Summer 1985

An Argus Specialist Publication

Personal SOFTWARE

£2.25

CBM 64 EXTENSIONS

OVER 20 GREAT UTILITIES:

Sprites * Graphics * Music * Sound * Memory & BASIC

VIC-20 BASIC enhanced

Educational fun for all the family

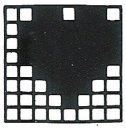
***Plus* games for the CBM64 and the unexpanded VIC-20**

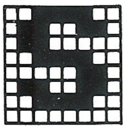
From the publishers of Your COMMODORE

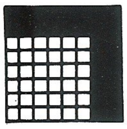
64 Symbols

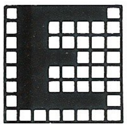
One of the commonest queries from 64 owners is the nature of the symbols found in some program listings. Since quite a number appear in this issue of **Personal Software,** we are including this page so you can understand them.

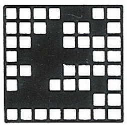
The 64 allows you to specify control keys, etc in print statements so that these functions can be executed within a program. For example, it is possible to position the cursor or clear the screen by using the relative symbol within quotation marks. The following list shows the symbols and the keys that generate them. We hope it's clearer now; but don't forget the quotes!

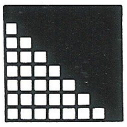

CLR

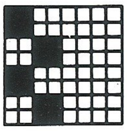

HOME

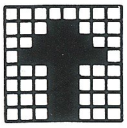

BLK

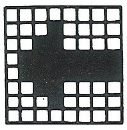

WHT

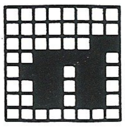

RED

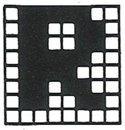

CYN

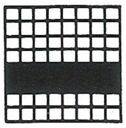

PUR

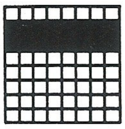

GRN

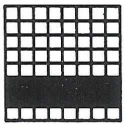

BLU

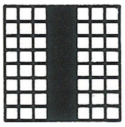

YEL

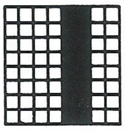

RVS ON

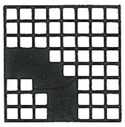

RVS OFF

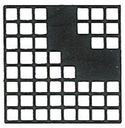

F1

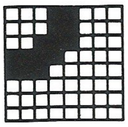

F3

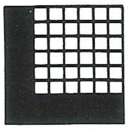

F5

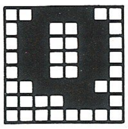

F7

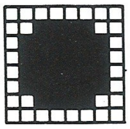

F2

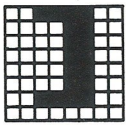

F4

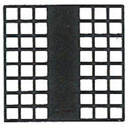

F6

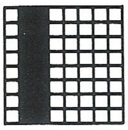

F8

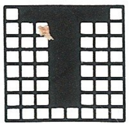

CURSOR
DOWN

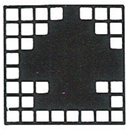

CURSOR
UP

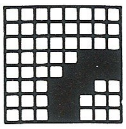

CURSOR
RIGHT

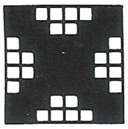

CURSOR
LEFT

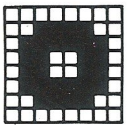

INSERT

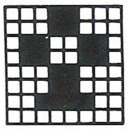

DELETE

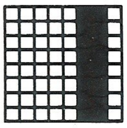

CBM & BLK

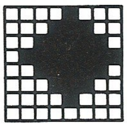

CBM & WHT

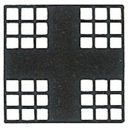

CBM & RED


CBM & CYN


CBM & PUR


CBM & GRN


CBM & BLU


CBM & YEL

Personal
SOFTWARE
CBM 64 EXTENSIONS
OVER 20 GREAT UTILITIES:
Sprites * Graphics * Music * Sound * Memory & BASIC
VIC-20 BASIC enhanced
Educational fun for all the family
Plus games for the CBM64 and the unexpanded VIC-20
From the publishers of Your Commodore

Volume 4
No 2
Summer 1985

Editor: Dave Reeder
Group Editor: Dave Bradshaw
Advertisement Manager: Julie Bates
Advertisement Copy Control: Lynn Collis
Chairman: Jim Connell
Design by Argus Design

Personal Software is a quarterly magazine appearing on the third Friday in May, August, November and February.

Distribution by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS Tel: 01-247 8233. Printing by: Alabaster Passmore & Sons Ltd, Tovil, Maidstone, Kent.

The contents of this publication including all articles, designs, plans, drawings and programs and all copyright and other intellectual property rights therein belong to Argus Specialist Publications Limited. All rights conferred by the Law of Copyright and other intellectual property rights and by virtue of international copyright conventions are specifically reserved to Argus Specialist Publications Limited and any reproduction requires the prior written consent of the Company. © 1985 Argus Specialist Publications Limited.

Subscription Rates: UK £10.60 for four issues including postage. Airmail and other rates upon application to Personal Software Subscriptions, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP1 1BB.

CONTENTS

Editorial & Advertisement Office
No 1 Golden Square, London W1R 3AB
Telephone: 01-437 0626 Telex: 8811896

64

Sprite Editor 4
Get the most from your sprite facilities with this neat utility

64

Extended BASIC for the 64 6
Major new commands to add to your machine with ease and style . . .

64

3D in Extended BASIC 16
. . . and even more to take Extended BASIC into the Third Dimension!

64

BASIC Disassembler 27
Delve into ROM and machine code routines with this invaluable utility.

64

Tales of the Windmill 30
Outwit the hungry rats in this fast-action machine code game.

64

User Defined Graphics 37
Sprites may be the flavour of the month, but UDGs can still show you a thing or two.

64

Code Saver 40
Save sprite data or any other memory areas to tape.

VIC 20

Galaxy Hero 41
These are the voyages of the Starship Exitprise and you have the helm.

VIC 20

The Bells! The Bells! 44
A chance to play at Quasimodo without having to look like him.

VIC 20

Music Aid 46
Making sweet music was never easier for budding maestros.

VIC 20

Collision Ahead 50
Eat the dots in your Formula 1 car, but keep an eye on the other crazy drivers!

VIC 20

VIC Enhancers 52
Time to be serious with these great graphic and screen size utilities.

VIC 20

Crazy Craneman 56
Ship to shore fun in a test of docking skills.

VIC 20

Great Britain 58
Learn more about user definable characters whilst drawing accurate maps.

64

The Little Professor 60
Can you have fun whilst learning arithmetic? You better believe it.

64

Great Graphics 65
A suite of memory-efficient routines to use alone or together.

64

Getting More from Your 64 68
A full, value-for-money feature to help you master the mysteries of sprites, hi-res graphics and sound.

64

64 Special 80
The hidden mode, hi-res graphics and multi-coloured flashing — all made easy at last.

64

Rocky IV 85
Ever wanted to punch your computer in frustration? We may have the answer for you.

64

Tape Filer 88
Get your cassettes into discs the simple way.

64

Music Maker 94
Don't get too excited at the chance of making music — you'll have to compose yourself first.

64

Sprite Dump 98
Lastly, a useful routine for printing out the sprites we've shown you how to create.

Credits: Our grateful thanks to the following individuals for their assistance in the production of this issue. Tony, Colin, Bill, Ed, Simon, Jane, Jamie, Jerry and others almost too numerous to mention.

TONY KELLY

Sprite Editor

Change the size, shape and colour of your sprites.

Although there are a number of graphics programs for the CBM64, I think my Sprite Editor program will be worth your time to type in as it helps you get the most out of the computer's sprite facilities.

It makes it easy to draw your sprite on screen and then view it in four different modes: normal, expanded, expanded in direction X only and expanded in direction Y only.

To move around you use the normal cursor keys plus colon, semicolon, full stop and slash. You can perform all the following functions:

Press 1: you will start printing as you move.

Press 2: you can move without deleting anything.

Press 3: you will delete the square you are on and any other you move to.

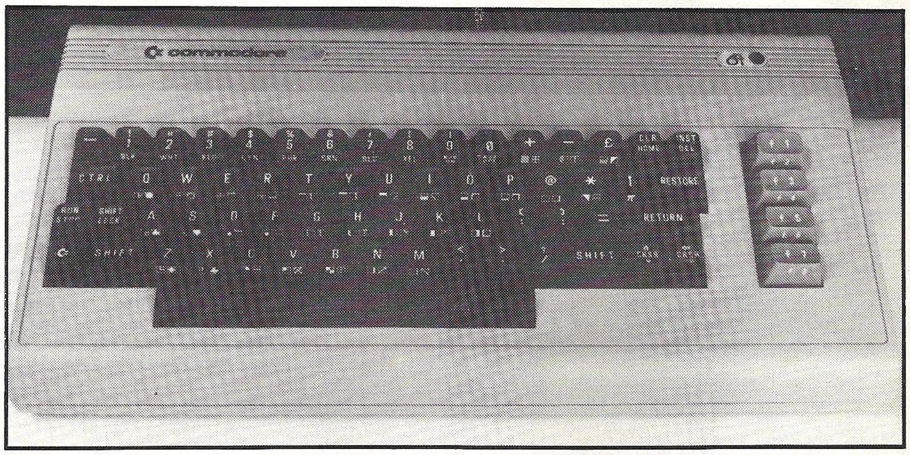
Press Q: changes the colour of the sprite to anything but black.

Press A: a check in case of wrong key press — you must answer Y (clears screen) or N (returns straight away).

Press S: press for data on right side of screen — pressing D will print data.

Press R: clears screen and returns to sprite draw.

Press any other key: your picture returns to the screen.



HOW IT WORKS

10-50	*sets variables and prints screen*
60-190	*reads keys pressed*
500-1170	*converts screen display into decimal code for sprites*
1190-1200	*POKEs sprite data into memory*
500-5100	*clears screen*
5500	*prints data*
5510	*prints last screen or runs program again*

VARIABLES USED

W$(21,24)	*holds screen picture*
F(62)	*holds sprite data*

```
9 REM [CLR]
10 PRINT"♥":DIMW$(21,24):POKE53281,0:POKE53280,0:DIMF(62):CO=0:V=53248
11 POKEV+21,15:POKEV+23,6:POKEV+29,10:POKEV,255:POKEV+1,175:POKEV+2,255
12 POKEV+3,130:POKEV+4,255:POKEV+5,80:POKEV+6,255:POKEV+7,200:POKE2040,200
14 POKE2041,200:POKE2042,200:POKE2043,200
19 REM[HOME]
20 PRINT"■";:PRINTCHR$(28);"╱────────────────────────╲"
30 FORA=0TO20:PRINT"▌"SPC(24);"▐":NEXT
40 PRINT"╲▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬╱"
50 DEFFNA(X)=1024+X+40*Y:X=1:Y=1
60 Z=PEEK(FNA(X))
66 IFPEEK(FNA(X))=32THENPOKEFNA(X),91:GOTO70
67 POKEFNA(X),219
70 GETA$:IFA$=""THEN70
72 IFP=1THENPOKEFNA(X),224
73 IFP=2THENPOKEFNA(X),32
74 IFP=0THENPOKEFNA(X),Z
```

```
75 REM[CRSR RIGHT]
80 IFA$="▐"THENX=X+1
81 IFA$="/"THENX=X+1:Y=Y+1
85 REM [CRSR LEFT]
90 IFA$="▐▌"THENX=X-1
91 IFA$=":"THENX=X-1:Y=Y-1
95 REM [CRSR DOWN]
100 IFA$="▓"THENY=Y+1
101 IFA$=";"THENY=Y-1:X=X+1
105 REM [CRSR UP]
110 IFA$="⌉"THENY=Y-1
111 IFA$="."THENY=Y+1:X=X-1
120 IFA$="1"THENP=1:POKEFNA(X),219
130 IFA$="2"THENP=0
140 IFA$="3"THENP=2:POKEFNA(X),32
145 IFA$="A"THENGOSUB5000
146 IFA$="Q"THENCO=CO+1:POKEV+40,CO:POKEV+39,CO:POKEV+41,CO:POKEV+42,CO
147 IFA$="D"THENPOKEV+21,0:GOSUB5500
149 IFCO=15THENCO=0
150 IFX>=24THENX=24
160 IFX<=1THENX=1
170 IFY<=1THENY=1
180 IFY>=21THENY=21
181 IFA$="S"THENPOKEV+21,0:GOSUB500
190 GOTO60
500 FORA=1TO21:FORB=1TO24
510 IFPEEK(1024+B+40*A)=32THENW$(A,B)="0":GOTO530
520 LETW$(A,B)="1"
530 NEXTB:NEXTA
1020 TH=0
1030 FORA=1TO21
1040 FORC=0TO2
1050 T=C*8
1060 SD=0
1070 IFW$(A,1+T)="1"THENSD=SD+128
1080 IFW$(A,2+T)="1"THENSD=SD+64
1090 IFW$(A,3+T)="1"THENSD=SD+32
1100 IFW$(A,4+T)="1"THENSD=SD+16
1110 IFW$(A,5+T)="1"THENSD=SD+8
1120 IFW$(A,6+T)="1"THENSD=SD+4
1130 IFW$(A,7+T)="1"THENSD=SD+2
1140 IFW$(A,8+T)="1"THENSD=SD+1
1150 F(TH)=SD:TH=TH+1
1160 NEXTC
1170 NEXTA
1190 FORA=0TO62:POKE200*64+A,F(A):NEXT
1210 POKEV+21,15:RETURN
4999 REM [HOME]
5000 PRINT"▣";SPC(26);"ARE YOU SURE":GETA$:IFA$=""THEN5000
5001 IFA$="Y"THENPRINT"▣";SPC(26);"         ":GOTO5009
5002 PRINT"▣";SPC(26);"         ":RETURN
5005 REM [BLU] [HOME] [CRSR RIGHT] [2 CRSR DOWNS]
5009 PRINT"▣▣▐▓▓                   ▣"
5010 FORA=1TO21:PRINT"▐                     ":NEXT:RETURN
5495 REM [CLR]
5500 PRINT"⌋";:FORA=0TO62:PRINTF(A),:NEXT
5510 GETA$:IFA$=""THEN5510
5520 IFA$="R"THENPOKE53281,7:POKE53280,7:RUN
5525 POKE53281,6:POKE53280,6:PRINT"⌋"
5530 FORA=1TO21:FORB=1TO24
5540 IFW$(A,B)="1"THENPOKE1024+B+40*A,224
5550 NEXTB:NEXTA:POKE53281,0:POKE53280,0
5560 POKEV+21,15: GOTO20
```

☐

DAVID REES

Extended BASIC for the 64

A whole host of major new commands for you to use.

Like to add many new commands to your 64? Although the DATA statements in these listings may look a little daunting, you will find the effort worthwhile to gain an Extended BASIC.

It is arranged in three sections: the DATA listings and definitions of the new commands and example programs.

Take care entering the machine code listings as any mistake can cause loss of your program. Always SAVE the program before you use the code. To help you find mistakes, a checksum is included — this adds up all the numbers in the DATA statements to show you if you have entered them correctly.

After the program has been POKEd into memory, you can dispense with the BASIC routine. You can use each section separately but the full potential of these routines only appears when you link them.

BASICs

The first listing gives the use of three commands: CLEAR GRAPHICS (CLG), POINT and LINE. For each command you must first POKE in parameters (no longer needed when you've linked to listing two) and then use SYS to run the command.

CLG is the easiest to use as it only has one parameter. Firstly, as with all the commands, we need to choose one of two hi-res screens: POKE 869,X does this (X=0 or 1 to choose screen) and then SYS 49879 runs the command.

POINT draws a point on either screen in normal or multicolour mode. First, you must POKE X and Y coordinates into memory — Y is easy as you just POKE a number between 0 and 199 to location 862; X has a range between 0 and 319 but two location are POKEd to, 861 for the first eight bits and 860 for the highest bit.

There are now four other registers to be set for POINT. Colour by POKEing 857 to the value required. To gain text mode, add the following command at the end of the program: POKE 56576,PEEK (56576)OR3. Next the colour choice (zero 0-1 in normal mode, zero 0-3 in multicolour) for the appearing colour to be stored in is POKEd to 858. Lastly, screen choice (0 or 1) is POKEd to 869 and Exclusive OR choice (on=1, off=0) is POKEd to 871.

This may seem complex but these four registers need only be set once. After this, you can draw as many points as you want to, changing registers only when you need to. After this, to draw the point, use SYS 49769.

LINE uses the final four registers of POINT in the same way, except viewed colour which must be renewed with each line, so that all that needs to be covered is X and Y choice.

Two positions need to be defined (for the two ends of the line) and you only need to POKE to the registers you are changing. The registers are: 1st point — 828 X high, 829 X low, 830 Y; 2nd point — 831 X high, 832 X low, 833 Y.

These are used in the same way as point registers. To draw the line use SYS 50544 but note that LINE is unreliable in multicolour mode.

Demo program 1 shows how to ise these commands and how to obtain screen 0; screen 1 can be obtained by adding POKEV+24,128:POKE 56576,PEEK(56576)AND252)+2.

Now type in machine code listing 1, RUN it and SAVE it And type NEW before entering Demo 1. Then SAVE and RUN.

DEFINITIONS

Now you've used the first section, you should be ready to get deeper into Extended BASIC. Listing 2 gives you more commands and Demo 2 examples of their use. As before use the checksum to check entry and SAVE before use, then see table for definitions for detail.

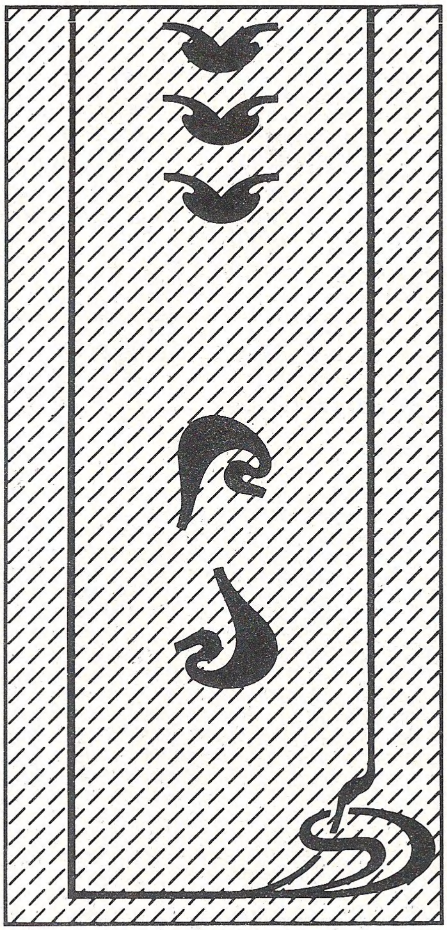
```
1 REM************************
2 REM*EXTENDED BASIC PRT 1*
3 REM*   BY DAVID REES     *
4 REM*        (1983)        *
5 REM*POINT+CLRG+LINE CODE*
6 REM* NO STANDARD BASIC  *
7 REM*        ENTRY         *
8 REM************************
20 T=0
30 FOR M=0 TO 2031
40 READA:POKE49152+M,A:T=T+A
50 NEXT
60 PRINT"CHECKSUM="T,"IT SHOULD BE 209186"
998 REM***** POINT *******
1000 DATA173,93,3,41,248,141,95,3,206,95,3
1010 DATA173,93,3,24,237,95,3,24,74,24,24
1020 DATA141,98,3,169,4,24,237,98,3
1030 DATA24,10,24,141,98,3,169,1,141,97,3
1040 DATA169,,205,98,3,208,1,96,206,98,3
1050 DATA14,97,3,24,144,241,96
1100 DATA173,93,3,41,248,141,95,3
1110 DATA173,93,3,24,237,95,3,24
1120 DATA141,98,3,169,7,24,237,98,3
1130 DATA24,141,98,3,169,1,141,97,3,169,
1140 DATA205,98,3,208,1,96,206,98,3
1150 DATA14,97,3,24,144,241,96
1200 DATA173,92,3,201,1,208,10
1210 DATA173,93,3,74,24,105,128,24
1220 DATA144,5,173,93,3,74,24,74,24
1230 DATA74,24,141,95,3
1240 DATA173,94,3,41,248,141,99,3
1250 DATA173,94,3,56,237,99,3,24
1260 DATA141,100,3,96
1300 DATA169,,133,252,173,95,3,160,
1310 DATA204,99,3,240,12,24,105,5
1320 DATA144,3,24,230,252,200,24
1330 DATA144,239,133,251,96
1400 DATA160,,132,254,173,100,3,24,101,251
1410 DATA144,3,24,230,254,200,192,8
1420 DATA208,243,133,253,165,254,160,
1430 DATA24,101,252,24,200,192,8
1440 DATA208,247,24,105,32,24
1450 DATA133,254,96
1500 DATA160,,173,22,208,41,16,201,
1510 DATA208,52,230,252,230,252,230,252
1520 DATA230,252,173,90,3,201,,208,12
1530 DATA177,251,41,240,24,109,89,3
1540 DATA24,145,251,96,177,251,41,15
1550 DATA24,14,89,3,14,89,3,14,89,3
1560 DATA14,89,3,24,109,89,3,24
```

```
1570 DATA145,251,96
1580 DATA173,90,3,201,,208,7,173,89,3
1590 DATA141,33,208,96,201,2,208,5
1600 DATA162,240,24,144,18,201,1,208,37
1610 DATA162,15,14,89,3,14,89,3,14,89,3
1620 DATA14,89,3,230,252,230,252
1630 DATA230,252,230,252,160,,177,251
1640 DATA45,13,3,24,109,89,3,24
1650 DATA145,251,96
1660 DATA173,101,3,201,,240,8,165,252
1670 DATA24,105,92,24,144,6,165,252
1680 DATA24,105,216,24,133,252
1690 DATA173,89,3,145,251,96
1800 DATA165,254,24,105,32,24,133,254
1810 DATA160,,173,22,208,41,16,201,
1820 DATA208,9,165,252,24,105,92,24
1830 DATA24,133,252,96
1840 DATA173,90,3,201,,208,1,96,201,3
1850 DATA208,1,96,165,252,24,105,4,24
1860 DATA133,252,96
2000 DATA173,103,3,201,1,240,40
2010 DATA173,102,3,201,1,240,21
2020 DATA169,255,24,237,97,3,141,104,3
2030 DATA160,,177,253,24,45,104,3,24
2040 DATA145,253,96,160,,177,253,24
2050 DATA13,97,3,24,145,253,96
2060 DATA160,,177,253,24,77,97,3,24
2070 DATA145,253,96
2100 DATA173,97,3,10,24,109,97,3,24
2110 DATA141,104,3,173,103,3,201,
2120 DATA240,9,160,,177,253,77,104,3
2130 DATA24,96,160,,169,255,24
2140 DATA237,104,3,24,141,104,3
2150 DATA173,90,3,201,,208,9,177,253
2160 DATA45,104,3,24,145,253,96
2170 DATA201,1,208,13,177,253
2180 DATA45,104,3,24,109,97,3,24
2190 DATA145,253,96,201,2,208,16
2200 DATA14,97,3,177,253,45,104,3
2210 DATA24,109,97,3,24,145,253,96
2220 DATA173,97,3,14,97,3,24,109,97,3
2230 DATA24,141,97,3,177,253,45,104,3
2240 DATA24,13,97,3,145,253,96
2300 DATA173,89,3,141,105,3
2305 DATA173,90,3,141,98,3,173,92,3
2310 DATA201,1,208,9,173,93,3,56
2320 DATA233,64,144,1,96,24,173,94,3
2330 DATA56,233,200,144,1,96,24
2340 DATA169,,141,88,3,173,22,208
2350 DATA41,16,201,,208,6,32,60,192
2360 DATA24,144,8,32,,192,169,1
2370 DATA141,88,3,32,112,192
2380 DATA32,161,192,32,190,192,173,101,3
2390 DATA201,,240,3,32,132,193,32,232,192
2395 DATA173,105,3,141,89,3
2400 DATA173,88,3,201,,240,4,32,242,193
2410 DATA96,32,183,193,96
2499 REM** CLS
2500 DATA169,32,162,64,172,101,3,192,
2505 DATA240,6,162,96,169,64,160,,133,252,169,
2510 DATA133,251,168,145,251,200
2520 DATA208,251,230,252,228,252
2530 DATA208,245,169,4,162,7,172,101,3
2535 DATA192,,240,6,162,99,169,96,160,
2540 DATA133,252,173,33,208,145,251,200
2550 DATA208,251,230,252,228,252
2560 DATA208,245,198,252,169,232
2570 DATA133,251,160,24,173,33,208,145,251
2580 DATA200,208,251,169,216,133,252
2590 DATA162,220,169,,168,133,251
2600 DATA145,251,200,208,251,230,252
2610 DATA228,252,208,245,96
2699 REM*** LINE ***
2700 DATA173,60,3,205,63,3,240,30,169,1
2710 DATA141,79,3,205,60,3,240,9,169,6
2720 DATA141,77,3,141,78,3,96,169,9
2730 DATA141,77,3,169,3,141,78,3,96
2740 DATA173,61,3,205,64,3,208,14
2750 DATA169,255,141,79,3,169,6
2760 DATA141,77,3,141,78,3,96,169,
2770 DATA141,79,3,173,61,3,56,237,64,3
2780 DATA144,12,24,169,9,141,77,3,169,3
2790 DATA141,78,3,96,169,6,141,77,3
2800 DATA141,78,3,96
```

```
2900 DATA162,,172,77,3,189,60,3
2910 DATA153,60,3,200,232,224,3,208,244
2920 DATA162,,172,78,3,189,63,3
2930 DATA153,63,3,200,232,224,3,208,244
2940 DATA173,79,3,201,1,240,17,169,
2950 DATA141,74,3,173,70,3,56,237,67,3
2960 DATA24,141,75,3,96,169,1,141,74,3
2970 DATA173,70,3,56,237,67,3,144,5
2980 DATA24,141,75,3,96,206,74,3,24
2990 DATA141,75,3,96
3000 DATA173,71,3,205,68,3,208,11,169,
3010 DATA141,76,3,169,2,141,80,3,96
3020 DATA56,237,68,3,24,141,76,3
3030 DATA173,71,3,56,237,68,3,144,7,24
3040 DATA169,1,141,80,3,96,169,,56
3050 DATA237,76,3,24,141,76,3,169,
3060 DATA141,80,3,96
3100 DATA173,80,3,201,2,208,6,169,1
3110 DATA141,81,3,96,173,74,3,201,1
3120 DATA208,6,169,1,141,81,3,96
3130 DATA173,75,3,56,237,76,3,144,7,24
3140 DATA169,1,141,81,3,96,169,
3150 DATA141,81,3,96
3200 DATA173,84,3,24,109,75,3,144,20,24
3210 DATA141,84,3,169,,238,83,3
3220 DATA205,83,3,208,9,238,82,3,24
3230 DATA144,3,141,84,3,173,83,3,24
3240 DATA109,74,3,144,4,24,238,82,3
3250 DATA141,83,3
3255 DATA96
3260 DATA173,87,3,24,109,77,3,144,4
3270 DATA24,238,86,3,141,87,3,173,86,3
3280 DATA24,109,76,3,24,141,86,3,96
3300 DATA173,87,3,56,237,77,3,176,3
3310 DATA206,86,3,24,141,87,3,173,86,3
3320 DATA56,237,76,3,24,141,86,3,96
3400 DATA173,75,3,201,,208,9,141,74,3,169,1
3410 DATA141,76,3,96,160,,140,112,3,173,76,3,200
3420 DATA24,109,76,3,144,249,24,238,112,3,170
3430 DATA173,112,3,56,237,75,3,176,4,138
3440 DATA24,144,231,24,234,140,75,3,169,1
3450 DATA141,76,3,169,,141,74,3,141,77,3,96
3500 DATA162,,173,76,3,201,,208,9,141,77,3,169,1
3510 DATA141,74,3,96,160,,140,112,3,173,74,3
3520 DATA201,1,208,2,162,1,173,75,3,200,24
3530 DATA109,75,3,144,15,24,238,112,3,72
3540 DATA173,112,3,56,237,76,3,176,22,104,224,1
3550 DATA208,230,238,112,3,72,173,112,3,56
3560 DATA237,76,3,176,4,104,24,144,213,24
3570 DATA234,104,140,77,3,169,1,141,74,3,169,
3580 DATA141,75,3,141,76,3,96
3799 REM*** MAIN LINE
3800 DATA32,66,195,32,161,195,32,246,195
3810 DATA32,49,196,173,66,3,141,82,3
3820 DATA173,67,3,141,83,3,173,68,3
3830 DATA141,86,3,169,,141,84,3
3840 DATA141,85,3,141,87,3,141,88,3
3850 DATA173,22,208,41,16,201,,240,5
3860 DATA169,1,141,88,3,173,74,3,205,76,3
3862 DATA208,27,173,75,3,205,77,3,208,19
3864 DATA169,,141,75,3,141,77,3
3866 DATA141,76,3,169,1,141,74,3
3868 DATA24,144,17
3869 DATA173,81,3,201,
3870 DATA208,6,32,202,196,24,144,3
3880 DATA32,14,197,24,173,83,3,141,93,3
3890 DATA173,82,3,141,92,3,173,86,3
3900 DATA141,94,3,32,105,194,173,97,3
3905 DATA141,112,3,32,172,199
3910 DATA173,81,3,201,,240,3
3920 DATA76,196,198,173,80,3,201,,240,3
3930 DATA76,47,199,76,102,199
4000 DATA173,82,3,141,92,3,174,83,3
4010 DATA32,97,196,173,83,3
4020 DATA236,83,3,208,1,96,141,93,3
4030 DATA78,97,3,24,173,97,3,201,
4040 DATA208,27,169,128,141,97,3,165,253,24
4050 DATA105,8,144,3,24,230,254,133,253
4060 DATA230,251,165,251,201,
4070 DATA208,2,230,252,96
4100 DATA173,86,3,141,94,3,32,146,196
4110 DATA173,86,3,205,94,3,208,1,96
4120 DATA141,94,3,41,7,201,,240,3
4130 DATA230,253,96,165,251,24,105,40
```

```
4140 DATA144,3,24,230,252,133,251
4150 DATA165,253,24,105,57,144,3,24
4160 DATA230,254,230,254,133,253,96
4200 DATA173,86,3,141,94,3,32,174,196
4210 DATA173,86,3,205,94,3,208,1,96
4220 DATA141,94,3,41,7,201,7,240,3
4230 DATA198,253,96,165,251,56,233,40
4240 DATA176,2,198,252,24,133,251
4250 DATA165,253,56,233,57,176,2
4260 DATA198,254,24,198,254,133,253,96
4300 DATA160,,32,43,198,173,93,3,24
4310 DATA105,1,144,4,24,238,92,3
4320 DATA141,93,3,173,80,3,201,1,208,6
4330 DATA32,82,198,24,144,3,32,139,198
4340 DATA173,94,3,201,255,208,1,96
4350 DATA201,200,208,1,96,173,92,3
4360 DATA201,,240,8,173,93,3,201,64
4370 DATA208,1,96,192,1,240,13,200
4380 DATA173,22,208,41,16,201,,240,3
4390 DATA76,198,198
4400 DATA32,157,199,32,200,194,173,92,3
4410 DATA205,69,3,208,9,173,93,3
4420 DATA205,70,3,208,1,96,76,196,198
4500 DATA238,94,3,173,94,3,32,103,198
4510 DATA32,19,198,173,94,3,201,255
4520 DATA208,1,96,201,200,208,1,96
4530 DATA173,92,3,201,,240,8,173,93,3
4540 DATA201,64,208,1,96
4550 DATA32,157,199,32,200,194
4560 DATA173,94,3,205,71,3,208,202
4570 DATA96
4600 DATA206,94,3,173,94,3,32,160,198
4610 DATA32,19,198,173,94,3,201,255
4620 DATA208,1,96,201,200,208,1,96
4630 DATA173,92,3,201,,240,8,173,93,3
4640 DATA201,64,208,1,96
4650 DATA32,157,199,32,200,194
4660 DATA173,94,3,205,71,3,208,202
4670 DATA96
4700 DATA160,,177,251,45,95,3,24
4710 DATA109,89,3,24,145,251,96
4720 DATA173,22,208,41,16,201,,208,30
4730 DATA173,90,3,201,,208,6,169,240
4740 DATA141,95,3,96,173,89,3,10,10,10
4750 DATA10,24,141,89,3,169,15,141,95,3
4760 DATA96,173,90,3,201,,240,11,201,2
4770 DATA240,222,201,1,208,218,24
4780 DATA144,221,169,255,141,95,3,169,
4790 DATA141,89,3,96
```

```
9 REM*DEMONSTRATION PROGRAM*
10 V=53248
19 REM*GET INTO MODE*
20 POKEV+24,31:POKEV+17,PEEK(V+17)OR32
29 REM*CLG*
30 POKE869,0:SYS49879
39 REM*POINT*
40 POKE858,1:POKE857,5
50 POKE870,1:POKE871,0
100 FORN=0TO319STEP2
110 X=N:Y=100-90*SIN(N/50.929)
120 POKE860,X/256:POKE861,XAND255:POKE862,Y
130 SYS49769
140 NEXT
199 REM*LINE
200 FORN=0TO199STEP4
210 POKE828,0:POKE829,10:POKE830,100
220 POKE831,1:POKE832,10:POKE833,N
230 POKE857,7:SYS50544
240 NEXT
250 POKE871,1:POKE858,0
260 FORN=0TO199STEP4
270 POKE828,1:POKE829,10:POKE830,100
280 POKE831,0:POKE832,10:POKE833,N
290 POKE857,2:SYS50544
300 NEXT
390 REM*BACK TO NORMAL*
400 GETA$:IFA$=""THEN400
410 POKEV+17,PEEK(V+17)AND223
420 POKEV+24,21
```



MODE

Allows easy change to graphics modes. The screen will only be cleared if you want it to be.
Format: MODE = 51828 SYSMODE,A,B,C

Values

A 0	*hi-res screen 0; 320x200 res; Teletext*
1	*hi-res screen 0; 160x200 res; N/A*
2	*extended colour; 40x25 res; 40x25 text*
3	*multicolour; 80x50 res; N/A*
4	*hi-res screen 1; 320x200 res; Teletext*
5	*hi-res screen 1; 160x200 res; N/A*
6	*normal graphics; 80x50 res; 40x24 text*
7	*normal text; 80x50 res; 40x25 text*
8	*user-def graphics; 40x25+ res; defined*
B: 0-1	*which hi-res screen is cleared*
C: 1	*to clear screen 1*

GCOL

Changes graphics colour.
Format: GCOL = 51770 SYSGCOL,A,B

Values:

A 0-3	*hi-res colour choice (on/off or multicolour)*
4	*text colour*
5	*border colour*
6	*background colour*
7-9	*colours 1-3 for ECM and multicolour modes*
10-1	*sprite multicolours 1 and 2*
12-9	*individual sprite colours*
B 0-15	*the colour put into these registers*

POINT

Draws a point in modes 0,1,4,5.
Format: PO=51334 SYSPO,A,X,Y

Values

A	**the attribute; each bit performs a separate job:**
0	**screen 0 if off; screen 1 drawn to if on**
1	**plot on if off; unplot if on**
2	**normal if off; exclusive/OR if on**
X,Y	**plot position (always 320x200)**

Plot colour is specified by SYSGCOL,(0-3),colour

GMOVE

Moves plotting position; for use with lines.
Format: GMOVE=52039 SYSGMOVE,X,Y
Values: X,Y is the plot position

LINE

Draws a line from last plotting position in modes 0,4.
Format: LINE=51754 SYSLINE,A,X,Y
Values: variable definitions as for POINT

TPOINT

Draws a point in text modes 3,6,7; plus 8 if graphics defined.
Format: TPO=51501 SYSTPO,A,X,Y
Values:

A 0/1	*normal or EXclusive OR*
X,Y	*plot position*

Plots in text modes in an 80x50 resolution, but for compatibility uses a 320x200 number grid

CLG

Clears graphics screen.
Format: CLG=52151 SYSCLG,A
Clears graphics screen denoted by A (0 or 1)

GTEXT

Prints one specified character given by a number(see manual page 132) in modes 0 to 4. Can be upper, lower, graphic, reverse field, single or double height with background and foreground colours independently defined. Also combined with hi-res graphics.
Format: GTEXT=52657 SYSGTEXT,X,Y,A,B,C
Values:

X,Y	*print position on a 40x25 grid*
A	*colour, given by background colour+16*text colour*
B	*bit 0 gives screen printed to;* *bit 1 determines single/double height text*
C	*character set: 0-255 for graphics; 265-511 for text*

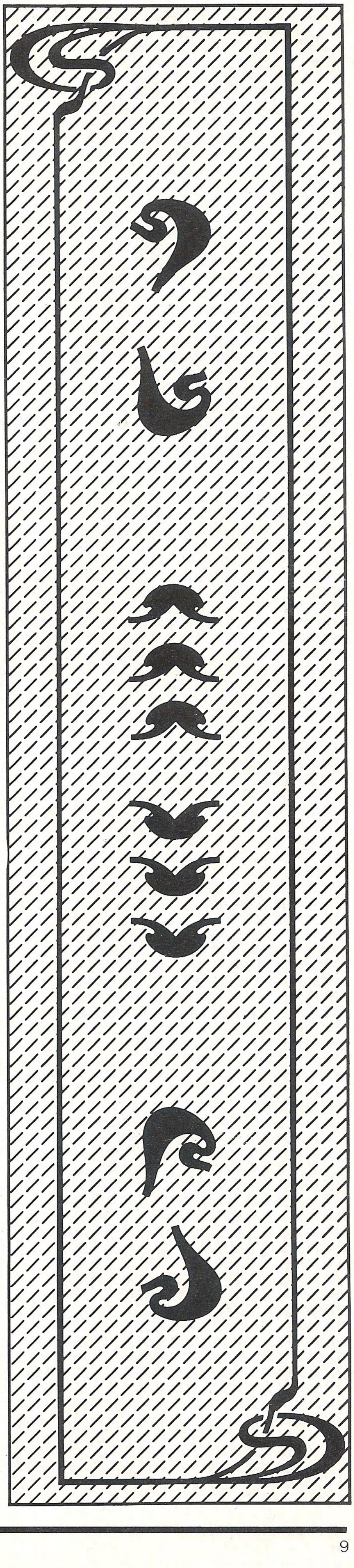
CURSOR

Set as printing position of the cursor relative to the top left hand corner of the screen.
Format: CRSR=53056 SYSCRSR,X,Y
Values: As for TAB(X,Y)

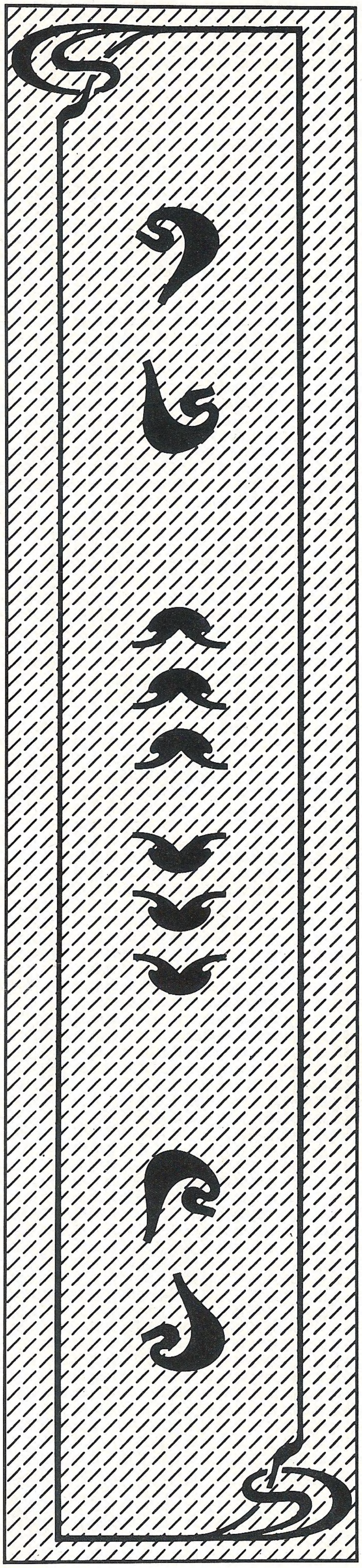
SPRITE

Changes the position of a sprite.
Format: SPRITE=52164 SYSSPRITE,N,X,Y
Values:

N 0-7	*sprite number*
X,Y	*position to maximum values 511,255*



▷



SETSPRITE

Defines video registers for sprites.
Format: SETSPRITE=52248 STSSETSPRITE,N,A,M,B,C,D,X,Y,P,G

N 0-7	*sprite number*
A 0-1	*sprite off/on*
M 0-1	*normal or multicolour*
B,C,D 0-15	*multicolours 0-2 for all sprites*
X 0-1	*expand X off/on*
Y 0-1	*expand Y off/on*
P 0-1	*foreground/sprite priority*
G 0-255	*start of sprite graphics*

MUSIC

Plays a constant musical note of independent volume.
Format: MUSIC=52453 SYSMUSIC,S,W,V,N,O,D
Values:

S 0-2	*sound channel*
W	*waveform (16,32,64 or 128)*
V 0-15	*volume independent of other channels*
N 0-11	*note number (C to B) — see manual page 153*
O 0-7	*octave number*
D	*dummy value for later expansion*

To cut note use zero volume or POKE waveform register to zero.

VOL

Sets master volume in range 0-15.
Format: VOL=52646 SYSVOL,V

FX

A few odd useful funtions.
Format: FX=53075 SYSFX,N,A
Values for N:

N A	*Function*
0 0-1	*changes display column number 38/40*
1 0-1	*changes display row number 24/25*
2 0-7	*fine scrolls screen Xwards*
3 0-7	*fine scrolls screen Ywards*
4 0-1	*blank screen or display*
5 0-255	*sets master volume and filter modes*
6 0-255	*sets IRQ speed (ie cursor and TI rate) — usually 67*
7	*sets repeat keys:A=0 cursor only; A=64 no keys; A=128 all keys*

PLOTTING RATE

The plotting rate for pixels per second, command only, is:
POINT-300; LINE-3500; TPOINT-1000

NEW MEMORY MAP

0	*systems variables*
1024	*screen and colour for hi-res screen 0*
2048	*BASIC RAM*
8192	*hi-res screen 0, continues to 16193*
14336	*user-defined graphics*
16384	*hi-res screen 1*
24576	*hi-res screen 1 colour, to 25575*
40960	*BASIC ROM*
49152	*Extended BASIC code to 53247*

COMMAND INPUT

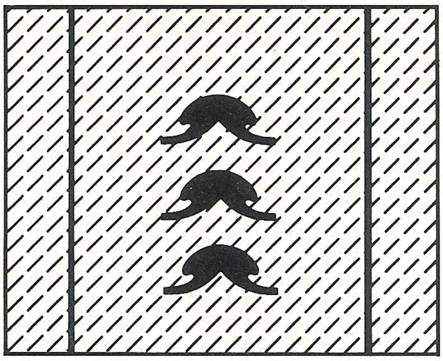
SYS to machine code. Location indicated by appropriately named variable — access time is saved by use of only the first two letters of this variable. Further values read by separate routine. This will read both variables and numbers but will not evaluate expressions. Only integer values can be used.

ERROR CALLS

Illegal Quality	*value out of range*
Syntax Error	*too many/few values after the command;* *if comma does not follow the command;* *if expression is used*

NOTE

Modes 4 and 5 use the second 16K block of memory, so sprite graphics must be POKEd to 16K and above, and registers normally at 2040 are moved to 25592.



```
1 REM*****************
2 REM*EXTENDED BASIC*
3 REM*   PART 2      *
4 REM*BY DAVID REES *
5 REM*****************
6 REM* TEXT,SPRITES *
7 REM*  AND SOUND    *
8 REM*****************
9 X=51184:C=0
10 FORN=0TO1788:READA:POKEX+N,A:C=C+A:NEXT
15 PRINT"CHECKSUM TOTAL 1="C
16 PRINT"IT SHOULD BE 205377■"
20 X=207*256:C=0
30 FORN=0TO226
40 READA:POKE(X+N),A:C=C+A
50 NEXT
55 PRINT"CHECKSUM TOTAL 2="C
56 PRINT"IT SHOULD BE 25677■"
100 DATA173,,221,41,252,24,105,3,24
110 DATA141,,221,32,129,255,162,14,108,,3
120 DATA108,10,3,96,32,4,200,165,102
125 DATA41,128,201,,240,3,76,240,199
130 DATA165,97,56,233,138,144,5,169,1
135 DATA133,253,96,169,,133,253
140 DATA133,252,133,251,165,97,201,
145 DATA208,3,133,251,96,41,127,201,1
150 DATA240,247,201,9,240,37
155 DATA168,169,128,136
160 DATA74,24,136,208,251,133,254
165 DATA169,1,133,20,165,98,37,254
170 DATA201,,240,6,165,251,5,20,133,251
175 DATA6,20,24,6,254,144,235,96
176 DATA169,1,133,252,165,98,10,24
177 DATA166,99,224,,240,3,24,105,1,24
178 DATA133,251,96
180 DATA32,8,200,165,253,240,3
185 DATA76,240,199,165,252,208,249,96
190 DATA32,8,200,165,253,240,3
195 DATA76,240,199,165,251,41,1
197 DATA141,101,3,165,251,41,2,240,2
200 DATA169,1,141,102,3,165,251,41,4
205 DATA240,2,169,1,141,103,3,173,22,207
210 DATA240,18,173,16,207,141,60,3
215 DATA173,17,207,141,61,3,173,18,207
220 DATA141,62,3,173,20,207,141,89,3
225 DATA173,21,207,141,90,3,32,8,200
230 DATA165,253,240,1,96,165,251
235 DATA141,64,3,141,93,3,141,17,207
240 DATA165,252,141,63,3,141,92,3
245 DATA141,16,207,32,8,200,165,253
250 DATA208,226
255 DATA165,252,208,222,165,251
260 DATA141,65,3,141,94,3,141,18,207
265 DATA173,19,207,240,13,201,1,240,9
270 DATA201,4,240,5,201,5,240,1,96
275 DATA173,22,207,240,15,160,
280 DATA140,22,207,201,2,240,3,76,107,203
285 DATA76,,,76,105,194
290 DATA32,119,200,165,251,41,1
295 DATA141,55,3,173,20,207,141,54,3
300 DATA32,8,200,165,253,240,1,96
305 DATA165,251,24,102,252,106,24
310 DATA102,252,106,24,141,52,3,56
315 DATA233,80,144,2,24,96,32,8,200
320 DATA165,253,240,1,96,165,252,240,1
325 DATA96,165,251,74,24,74,24,141,53,3
330 DATA56,233,50,144,2,24,96
335 DATA173,19,207,201,3,240,13,201,6
340 DATA240,9,201,7,240,5,201,8,240,1
345 DATA24,173,52,3,24,74,24,141,56,3,10
350 DATA24,141,57,3,173,52,3,56
355 DATA237,57,3,24,141,60,3,173,53,3
360 DATA24,74,24,141,58,3,10,24
365 DATA141,59,3,173,53,3,56,237,59,3
370 DATA24,10,24,141,61,3,24,109,60,3
375 DATA24,168,169,1,192,,240,8,24,10
380 DATA24,136,192,,208,248,141,62,3
385 DATA169,4,133,252
390 DATA173,56,3,172,58,3,192,,240,13
395 DATA24,105,40,144,3,24,230,252,136
400 DATA192,,208,243,133,251,160,
405 DATA177,251,217,,207,240,7,200
410 DATA192,16,208,246,160,,152
415 DATA172,55,3,192,,208,6,13,62,3
420 DATA24,144,3,77,62,3,168,185,,207
425 DATA160,,145,251,165,252,24
430 DATA105,212,24,133,252,173,54,3
435 DATA145,251,96
440 DATA169,1,141,22,207,76,134,200
445 DATA169,2,173,22,207,76,134,200
450 DATA32,119,200,165,251,133,2
455 DATA32,119,200,166,251,165,2,56
460 DATA233,4,176,9,142,20,207,165,2
465 DATA141,21,207,96,24,201,,208,4
470 DATA142,134,2,96,56,233,1,24
475 DATA133,2,56,233,15,144,2,24,96
480 DATA164,2,165,251,153,32,208,96
485 DATA32,119,200,173,,221,41,252,9,3
490 DATA141,,221,173,17,208,41,159
495 DATA141,17,208,173,22,208,41,239
500 DATA141,22,208,169,21,141,24,208
505 DATA165,251,56,233,9,144,4,24
510 DATA76,240,199,165,251,133,2
515 DATA32,8,200,165,251,41,1,141,101,3
520 DATA32,8,200,165,251,41,1,201,
525 DATA240,3,32,215,194,165,2
530 DATA141,19,207,201,,208,14,169,31
535 DATA141,24,208,173,17,208,9,32
540 DATA141,17,208,96,201,1,208,11
545 DATA173,22,208,9,16,141,22,208,24
550 DATA144,227,201,2,208,14,169,21
555 DATA141,24,208,173,17,208,9,64
560 DATA141,17,208,96,201,3,208,14
565 DATA169,21,141,24,208,173,22,208
570 DATA9,16,141,22,208,96,201,4
575 DATA208,24,169,128,141,24,208
580 DATA173,,221,41,252,9,2,141,,221
585 DATA173,17,208,9,32,141,17,208,96
590 DATA201,5,208,11,173,22,208,9,16
595 DATA141,22,208,24,144,217,201,6
600 DATA208,1,96,201,7,208,6,169,23
605 DATA141,24,208,96,169,31,141,24,208
610 DATA96
615 DATA32,8,200,165,253,208,11,234
620 DATA165,252,141,16,207,165,251
625 DATA141,17,207,32,8,200,165,253
630 DATA240,1,96,165,252,208,251
635 DATA165,251,141,18,207,96
640 DATA173,62,3,205,65,3,208,17
645 DATA173,61,3,205,64,3,208,9
650 DATA173,60,3,205,63,3,208,1,96
655 DATA173,62,3,56,233,200,144,11,24
660 DATA173,65,3,56,233,200,144,2,24,96
662 DATA173,60,3,240,24,173,63,3,240,19
```

```
665 DATA173,61,3,56,233,64,144,11,24
667 DATA173,64,3,56,233,64,144,2,24,96
668 DATA76,112,197
670 DATA32,119,200,165,251,41,1
675 DATA141,101,3,76,215,194
680 DATA32,119,200,165,251,41,7,24,10
685 DATA24,133,2,169,1,133,255,164,251
690 DATA192,,240,6,6,255,24,136,208,250
695 DATA32,8,200,165,253,240,3
700 DATA76,240,199,165,251,164,2
705 DATA153,,208,165,252,240,8
710 DATA173,16,208,5,255,24,144,13
715 DATA169,255,56,229,255,24,133,255
720 DATA173,16,208,37,255,141,16,208
725 DATA32,119,200,165,251,164,2
730 DATA153,1,208,96
735 DATA32,119,200,165,251,41,7,133,2
740 DATA169,1,133,255,164,251,192,
745 DATA240,6,6,255,24,136,208,250
750 DATA169,255,56,229,255,24,133,75
755 DATA32,119,200,165,251,41,1,201,
760 DATA240,8,173,21,208,5,255,24
765 DATA144,5,173,21,208,37,75
770 DATA141,21,208,32,119,200,165,251
775 DATA41,1,201,,240,8,173,28,208
780 DATA5,255,24,144,5,173,28,208,37,75
785 DATA141,28,208,32,119,200,165,251
790 DATA164,2,153,39,208,32,119,200
795 DATA165,251,141,37,208,32,119,200
800 DATA165,251,141,38,208,32,119,200
805 DATA165,251,41,1,201,,240,8
810 DATA173,29,208,5,255,24,144,5
815 DATA173,29,208,37,75,141,29,208
820 DATA32,119,200,165,251,41,1,201,
825 DATA240,8,173,23,208,5,255,24
830 DATA144,5,173,23,208,37,75
835 DATA141,23,208,32,119,200,165,251
840 DATA41,1,201,,240,8,173,27,208
845 DATA5,255,24,144,5,173,27,208
850 DATA37,75,141,27,208,32,119,200
855 DATA165,251,164,2,153,248,7,96
955 DATA32,119,200,165,251,41,3,201,3
960 DATA208,1,96,168,192,,240,10,169,
965 DATA24,105,7,24,136,208,249,168
970 DATA132,2,32,119,200,165,251,9,1
975 DATA133,255,41,254,164,2,153,4,212
980 DATA169,15,153,5,212,32,119,200
985 DATA165,251,41,15,10,24,10,24,10,24
990 DATA10,24,164,2,153,6,212,32,8,200
995 DATA165,253,240,1,96,165,252
1000 DATA208,251,165,251,56,233,12
1005 DATA176,244,24,165,251,10,24,168
1010 DATA185,40,207,133,80,185,41,207
1015 DATA133,79,32,119,200,165,251
1020 DATA41,7,133,81,169,7,56,229,81
1025 DATA24,168,201,,240,12,165,80
1030 DATA24,106,102,79,24,136,208,248
1035 DATA133,80,32,8,200,165,253,240,3
1040 DATA76,240,199,165,252,133,82
1045 DATA165,251,24,42,38,82,24,42
1050 DATA38,82,24,164,2,153,23,207
1055 DATA165,82,153,24,207,169,1
1060 DATA153,25,207,165,79,153,,212
1065 DATA165,80,153,1,212,165,255
1070 DATA153,4,212,96
1075 DATA32,119,200,165,251,41,15
```

```
1080 DATA141,24,212,96
1100 DATA32,119,200,165,251,56,233,40
1110 DATA144,4;24,76,240,199,165,251,141,70,3
1120 DATA32,119,200,165,251,56,233,25
1130 DATA176,237,165,251,141,71,3,32,119,200
1140 DATA165,251,141,72,3,32,119,200,165,251
1150 DATA41,1,141,73,3,165,251,41,2
1160 DATA141,74,3,201,,240,10,173,71,3
1170 DATA201,24,208,3,76,240,199
1180 DATA32,8,200,165,251,24,42,38,252
1190 DATA24,42,38,252,24,42,38,252,24
1200 DATA133,251,165,252,24,105,208,24
1210 DATA133,252,169,4,133,81,173,70,3
1220 DATA172,71,3,192,,240,11,24
1230 DATA105,40,144,3,24,230,81,136
1240 DATA208,245,133,80,173,73,3,240,8
1250 DATA165,81,24,105,92,24,133,81
1260 DATA169,,133,254,173,70,3,24,42
1270 DATA38,254,24,42,38,254,24,42,38,254
1280 DATA24,172,71,3,192,,240,13,24
1290 DATA105,64,144,3,24,230,254
1300 DATA230,254,136,208,243,133,253
1310 DATA173,73,3,208,8,165,254,24
1320 DATA105,32,24,144,5,165,254,24
1330 DATA105,64,24,133,254,173,14,220
1340 DATA41,254,141,14,220,165,1,41,251
1350 DATA133,1,160,,173,74,3,208,29
1360 DATA173,72,3,145,80,177,251,145,253
1370 DATA200,192,8,208,247,165,1,9,4
1380 DATA133,1,173,14,220,9,1,141,14,220,96
1390 DATA173,72,3,145,80,160,40,145,80
1400 DATA169,,162,,133,82,133,83
1410 DATA164,82,177,251,164,83,145,253
1420 DATA200,145,253,230,82,230,83
1430 DATA230,83,192,7,208,235,165,253
1440 DATA24,105,64,144,3,24,230,254
1450 DATA230,254,133,253,232,169,,224,2
1460 DATA208,212,24,144,179
2000 DATA32,126,124,226,123,97,255,236
2010 DATA108,127,225,251,98,252,254,160
2020 DATA,,,6,,,,,,,,49,234, ,,,,,,,,,,
2030 DATA134,30,142,24,150,139,159,126
2040 DATA168,250,179,6,189,172,200,243
2050 DATA212,230,225,143,238,248,253,46
2100 DATA32,119,200,165,251,133,80
2110 DATA32,119,200,165,251,24,170
2120 DATA164,80,76,240,255
2130 DATA32,119,200,165,251,41,7,133,81
2140 DATA32,119,200,165,251,170,165,81
2150 DATA208,18,138,240,2,169,8,133,82
2160 DATA173,22,208,41,247,5,82
2170 DATA141,22,208,96,201,1,208,18
2180 DATA138,240,2,169,8,133,82
2190 DATA173,17,208,41,247,5,82
2200 DATA141,17,208,96,201,2,208,16
2210 DATA138,41,7,133,82,173,22,208
2220 DATA41,248,5,82,141,22,208,96
2230 DATA201,3,208,16,138,41,7,133,82
2240 DATA173,17,208,41,248,5,82
2250 DATA141,17,208,96,201,4,208,18
2260 DATA138,240,2,169,16,133,82
2270 DATA173,17,208,41,239,5,82
2280 DATA141,17,208,96,201,5,208,5
2290 DATA138,141,24,212,96,201,6,208,5
2300 DATA138,141,5,220,96,138,141,138,2
2310 DATA96
```

```
10 MODE=51828:GCOL=51770:PO=51334
20 GMOVE=52039:LINE=51754:TPO=51501
30 CLG=52151:GTEXT=52657:CRSR=53056
40 SPRITE=52164:SETSPRITE=52248
50 MUSIC=52453:VOL=52646:FX=53075
80 SYSGCOL,5,2:SYSGCOL,4,6
98 REM*IF YOU DO NOT HAVE PART ONE*
99 REM*GOTO290*
100 SYSMODE,0,0,1
110 SYSGCOL,1,0
120 FORN=0TO319STEP2
130 Y%=100-80*SIN(N*π/160):SYSPO,2,N,Y%
140 NEXT
150 SYSGCOL,1,7
160 FORN=0TO199STEP6
170 SYSGMOVE,30,100:SYSLINE,2,300,N
180 NEXT
190 SYSGCOL,0,8
200 FORN=0TO199STEP6
210 SYSGMOVE,310,100:SYSLINE,6,30,N
220 NEXT
290 SYSMODE,0,0,0
300 A(0)=8:A(1)=5:A(2)=12:A(3)=12:A(4)=15
310 FORN=0TO4
320 SYSGTEXT,N,0,235,0,A(N)
330 SYSGTEXT,N,1,5,2,A(N)
340 NEXT
400 FORN=0TO4
410 X=N+5:C=A(N)+256:SYSGTEXT,X,0,114,0,C
420 SYSGTEXT,X,1,39,2,C
```

```
430 NEXT
500 FORN=0TO4
510 X=N+10:C=A(N)+128:SYSGTEXT,X,0,16,0,C
520 SYSGTEXT,X,1,1,2,C
530 NEXT
590 FORN=0TO999:NEXT
600 SYSMODE,6,0,0:PRINT"ℷ"
610 SYSGCOL,5,0:SYSGCOL,6,0
620 SYSGCOL,1,4
630 FORN=4TO315STEP4:SYSTPO,0,N,100:NEXT
640 SYSGCOL,1,8
650 FORN=0TO199STEP4
660 SYSTPO,0,4,N:SYSTPO,0,315,N
670 NEXT
680 SYSGCOL,1,6
690 FORN=8TO311STEP4
700 S=100-90*SIN((N-8)*π/152):SYSTPO,0,N,S
710 NEXT
720 SYSGCOL,1,2
730 FORN=8TO311STEP4
740 S=100-90*COS((N-8)*π/152):SYSTPO,0,N,S
750 NEXT
790 FORN=0TO999:NEXT
800 PRINT"ℷ"
810 FORN=0TO39
820 Y%=12-12*SIN(N*π/20):SYSGCOL,4,N
830 SYSCRSR,N,Y%:PRINT"HELLO"
840 NEXT
920 M=15
925 X=32
930 FORO=7TO2STEP-1
940 FORS=23TO0STEP-2:T=5/(S+3)
944 SYSMUSIC,0,X,0,0,O,0
945 L=O-1:SYSMUSIC,1,X,0,0,L,0
947 FORTT=0TO99:NEXT
950 K%=S/2:SYSMUSIC,0,X,15,K%,O,0
951 L=O-1:SYSMUSIC,1,X,15,K%,L,0
970 FORN=0TO15STEPT:V%=15-N:SYSVOL,V%:NEXTN,S,O
980 SYSVOL,0
```

EXAMPLES

Finally, we can use our Extended BASIC in some sample programs to show how to use some of the commands easily. Hopefully, these will give you many ideas for possible uses of these powerful new commands.

POLAR GRAPHS

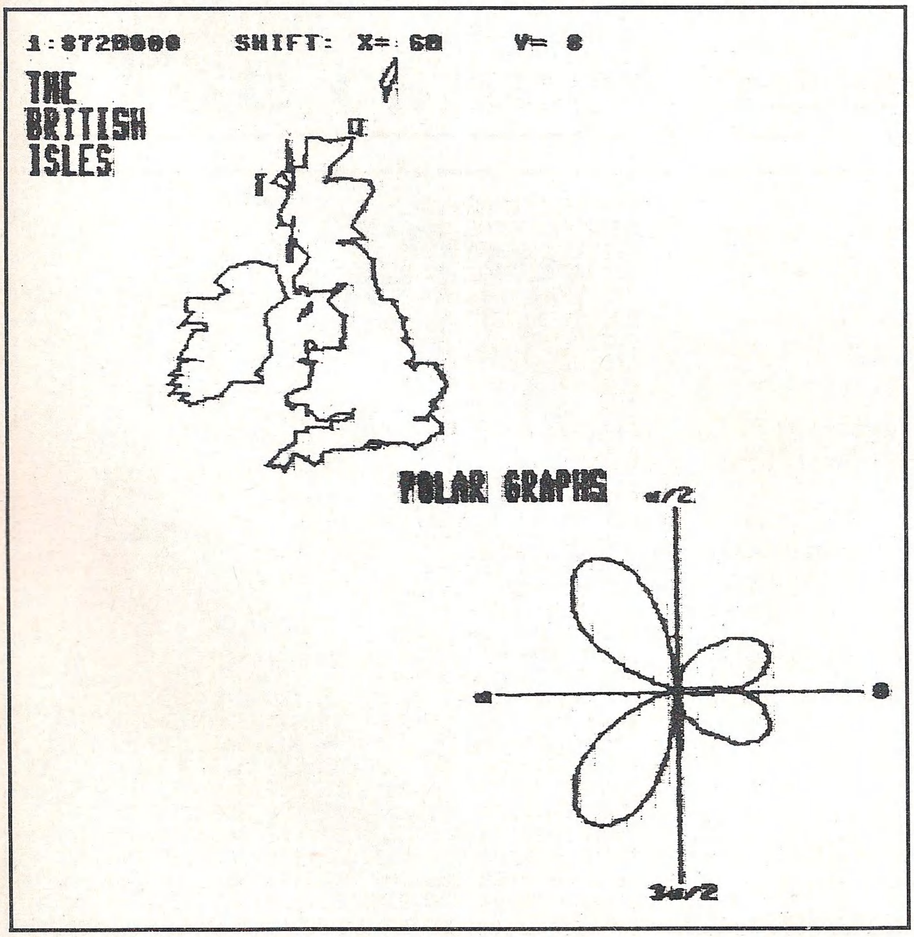
This will draw most polar functions at any magnification from a range of zero to 2pi. Apart from its educational value, it is also the easiest way to draw a good heart on a computer!

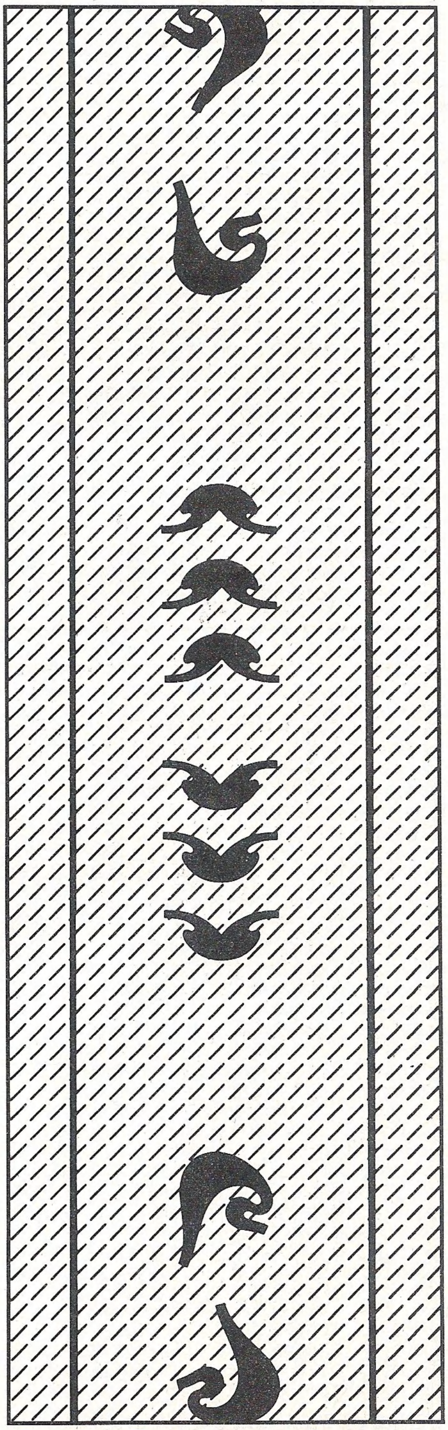
TELETEXT

This draws a replica of the Ceefax-type screen to show the versatility of Extended BASIC and how it can be used well.

BRITISH ISLES

Perhaps the most useful of the demo programs. This provides an accurate (within 1% of error) database for drawing maps of this area. It would be fairly simple to extend this to Europe.



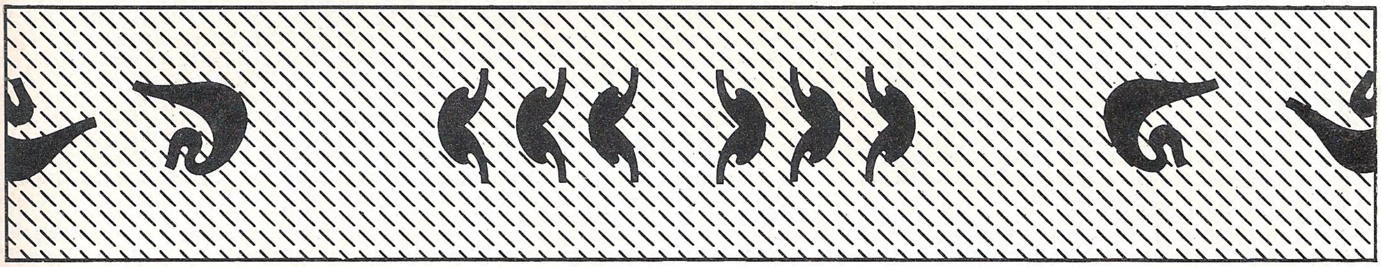


```
5 REM*POLAR GRAPH DEMONSTRATION*
6 REM*       BY DAVID REES       *
10 MODE=51828:GCOL=51770:PO=51334
20 GMOVE=52039:LINE=51754:GTEXT=52657
30 SYSGCOL,5,0:SYSGCOL,6,0
100 SYSMODE,0,0,1
110 SYSGCOL,1,1
120 SYSGMOVE,159,17
130 SYSLINE,2,159,183
140 SYSGMOVE,76,99
150 SYSLINE,2,242,99
197 REM* M SETS RADIUS OF 1 UNIT *
198 REM*IF THE GRAPH WILL NOT FIT*
199 REM* ON THE SCREEN REDUCE M  *
200 M=25
210 Y=99-M:SYSPO,2,158,Y:Y=99+M:SYSPO,2,160,Y
220 X=159-M:SYSPO,2,X,100:X=159+M:SYSPO,2,X,98
300 SYSGTEXT,31,12,230,0,48
310 SYSGTEXT,8,12,230,0,94
320 SYSGTEXT,18,1,230,0,94
325 SYSGTEXT,19,1,230,0,47
330 SYSGTEXT,20,1,230,0,50
340 SYSGTEXT,18,23,230,0,51
345 SYSGTEXT,19,23,230,0,94
350 SYSGTEXT,20,23,230,0,47
355 SYSGTEXT,21,23,230,0,50
400 A$=" POLAR GRAPHS "
410 FORN=0TO13
420 C$=MID$(A$,N+1,1):C=ASC(C$)
430 C=C+64*((C>64)+(C>128))
440 SYSGTEXT,N,0,226,2,C
450 NEXT
899 REM*DEFINE THIS FUNCTION*
900 DEFFNX(X)=2*SIN(2*X)-SIN(3*X)
950 L=80/M:N=0
960 R=FNX(N):S=SGN(R):A=ABS(R)
970 IFA>LTHENR=L*S
980 X%=R*M*SIN(N)+159:Y%=R*M*COS(N)+99
990 SYSGMOVE,X%,Y%
1000 FORN=0TO2.001*πSTEPπ/32
1010 R=FNX(N):S=SGN(R):A=ABS(R)
1020 IFA>LTHENR=L*S
1030 X%=R*M*SIN(N)+159:Y%=R*M*COS(N)+99
1040 SYSLINE,2,X%,Y%
1050 NEXT
1100 GETA$:IFA$=""THEN1100
1110 SYSMODE,6,0,0:PRINT"⊐"
```

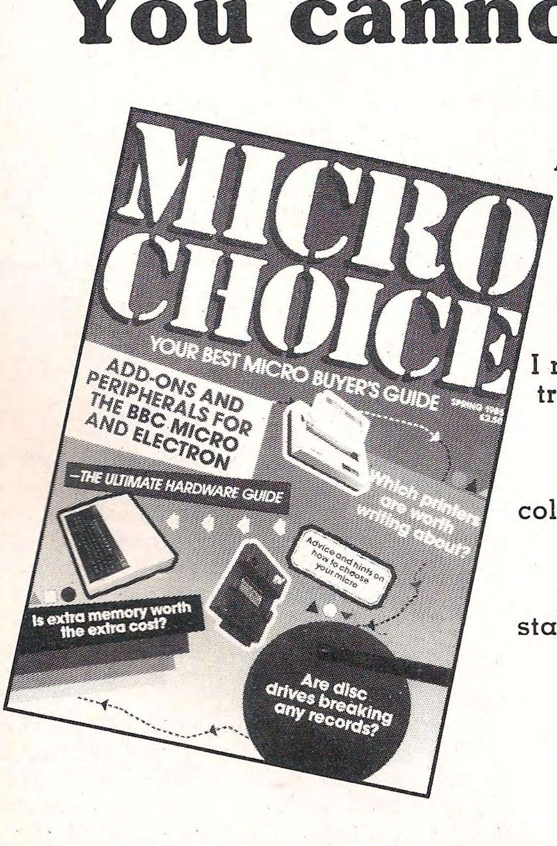
```
2 REM* DEMONSTRATION OF TELETEXT*
10 MODE=51828:GCOL=51770:GMOVE=52039
20 LINE=51754:GTEXT=52657
30 POKE51,0:POKE52,31:POKE55,0:POKE56,31
40 SYSGCOL,5,0:SYSGCOL,6,0
100 SYSMODE,0,0,1
110 A$="  ⌉198 64´—♣♣ 000  |UE 27 ‾EC  "
120 C=16:X=0:Y=0:H=0:CH=1:R=0:GOSUB1000
200 A$=LEFT$(TI$,2)+":"+MID$(TI$,3,2)+"/"
210 A$=A$+RIGHT$(TI$,2):C=80:X=32
220 Y=0:GOSUB1000
300 A$=" THIS IS COMMODORE 64 TELETEXT "
310 C=46:Y=2:X=4:H=1:CH=0:R=1:GOSUB1000
400 A$=" ♣ NEW MODE FOR THE COMPUTER "
410 C=110:Y=5:X=5:H=0:CH=1:R=1:GOSUB1000
500 A$="‾EVELOPED BY ‾♣×⌝‾ _‾‾♦ USING THE NEW"
510 C=224:Y=7:X=1:H=0:CH=1:R=0:GOSUB1000
600 A$="‾XTENDED |ASIC PROGRAM IN |OME"
610 A$=A$+" ‾OMPUTING"
620 C=224:Y=8:X=0:H=0:CH=1:R=0:GOSUB1000
650 A$="oEEKLY."
660 C=224:Y=9:X=0:H=0:CH=1:R=0:GOSUB1000
700 A$="  \ORE FROM 64´—♣♣ IF YOU PROGRAM IT "
710 A$=A$+"!  "
720 C=114:Y=24:X=0:H=0:CH=1:R=0:GOSUB1000
790 C=80:X=32:Y=0:CH=0:H=0:R=0
800 K=V:V=INT(VAL(TI$)/60)
810 IFK=VTHEN880
820 A$=LEFT$(TI$,2)+":"+MID$(TI$,3,2)
830 X=32:GOSUB1000
880 A$=RIGHT$(TI$,2):X=38:GOSUB1000
900 GETA$:IFA$=""THEN800
909 REM*...[CLR HOME]*
910 SYSMODE,6,0,0:PRINT"⊐"
998 END
999 REM*PRINT ROUTINE*
1000 A=INT((SAND1)+2*(HAND1)):L=LEN(A$):Y=INT(Y)
1030 FORN=1TOL:B$=MID$(A$,N,1):B=ASC(B$)
1040 D=B+32*(2*(B>63)+2*(B>191)):X%=X-1+N
1050 D=D+128*R+256*CH:SYSGT,X%,Y,C,A,D:NEXT
1060 RETURN
```

```
9 REM*BRITISH ISLES"
10 MODE=51828:GCOL=51770:GMOVE=52039
20 LINE=51754:GTEXT=52657
30 SC=4360000:E=319:F=199:G=-10:H=340:I=220
100 SYSGCOL,5,0:SYSGCOL,6,0
110 SYSMODE,6,0,0
120 PRINT"⊐"
130 INPUT"MAGNIFICATION (0.1-9.9)";M
140 IFM<0.1ORM>9.9THEN130
150 M=INT(M*10)/10
160 PRINT"▓PLEASE ENTER START CO-ORDINATES"
170 PRINT"IN A RANGE BETWEEN -4000 AND 4000"
180 INPUT"X";CX
190 IFCX<-4000ORCX>4000THEN180
200 CX=INT(CX)
210 INPUT"Y";CY
220 IFCY<-4000ORCY>4000THEN210
230 CY=INT(CY)
400 SYSMODE,4,1,1
405 FORMM=0TO7
410 READX,Y:X%=X*M+CX:Y%=Y*M+CY:IFX=-1THEN610
420 R=(X%<G)+(Y%<G)+(X%>H)+(Y%>I)
430 IFR<>0THEN410
440 IFX%>ETHENX%=E
450 IFX%<0THENX%=0
460 IFY%<8THENY%=8
470 IFY%>FTHENY%=F
480 SYSGMOVE,X%,Y%
500 FORN=0TO999
510 READX,Y:X%=X*M+CX:Y%=Y*M+CY:IFX=-1THEN610
520 R=(X%<G)+(Y%<G)+(X%>H)+(Y%>I)
530 IFR<>0THENK=1
540 IFX%>ETHENX%=E
550 IFX%<0THENX%=0
560 IFY%<8THENY%=8
570 IFY%>FTHENY%=F
580 IFK=1THENK=0:SYSGMOVE,X%,Y%:GOTO600
590 SYSLINE,3,X%,Y%
600 NEXTN
610 NEXTMM
620 V=INT(SC/M/10000)*10000
800 A$="1:"
810 B$=STR$(V):L=LEN(B$):A$=A$+RIGHT$(B$,L-1)
820 X=0:GOSUB900
830 A$="SHIFT: X="+STR$(INT(CX))
840 X=16:GOSUB900
850 A$="Y="+STR$(INT(CY))
860 X=32:GOSUB900
870 GETA$:IFA$=""THEN870
880 SYSMODE,6,0,0:PRINT"⊐"
899 END
900 L=LEN(A$):C=230
910 FORN=1TOL:C$=MID$(A$,N,1):D=ASC(C$)
920 IFD>63THEND=D-64
930 X%=X+N-1:SYSGTEXT,X%,0,C,1,D:NEXT:RETURN
1000 DATA300,325,310,325,315,320,325,330
1010 DATA325,337,315,340,314,343,300,350
1020 DATA290,345,275,347,260,343,262,345
1030 DATA267,349,262,351,252,349,262,345
1040 DATA260,343,250,350,230,350,215,355
1050 DATA214,360,208,365,200,360,190,355
1060 DATA180,370,175,365,170,367,165,365
1070 DATA180,350,190,345,192,335,200,335
1080 DATA200,330,225,332,225,325,235,324
```

```
1090 DATA240,322,245,314,240,318,226,320
1100 DATA221,325,213,325,207,317,200,320
1110 DATA202,316,200,310,193,311,189,315
1120 DATA187,315,182,310,185,308,182,302
1130 DATA205,295,207,285,207,274,209,274
1140 DATA209,273,210,272,208,273,207,272
1150 DATA206,273,200,275,195,273,202,270
1160 DATA205,265,200,265,200,255,205,255
1165 REM
1170 DATA210,260,205,265,229,260,231,262
1180 DATA231,260,233,261,234,260,232,258
1190 DATA235,255,233,253,235,250,237,232
1200 DATA235,233,222,219,237,205,232,204
1210 DATA230,208,229,213,226,212,215,210
1220 DATA210,210,205,215,198,207,195,213
1230 DATA190,200,205,183,192,170,192,165
1240 DATA191,165,191,170,188,185,185,184
1250 DATA186,170,193,145,189,152,175,155
1260 DATA180,150,178,145,176,146,186,125
1270 DATA184,120,170,115,180,105,190,120
1280 DATA190,100,200,100,200,90,202,90
1290 DATA205,75,225,79,245,75,241,85
1300 DATA225,100,227,105,220,115,235,110
1310 DATA242,112,260,112,265,120,243,155
1320 DATA253,165,231,169,235,170,242,166
1330 DATA263,185,265,200,270,218,280,221
1340 DATA290,235,292,236,291,237,295,255
1350 DATA290,253,300,265,300,270,293,275
1360 DATA300,280,305,275,320,276,327,284
1370 DATA330,293,326,300,325,305,317,309
1380 DATA319,311,317,313,321,315,310,322
1390 DATA300,325,-1,-1
1400 DATA195,236,200,227,207,220,207,229
1410 DATA195,235,-1,-1
1420 DATA187,104,190,96,187,92,187,85
1430 DATA185,77,185,87,187,104,-1,-1
1440 DATA157,107,165,106,161,115,157,107,-1,-1
1450 DATA157,115,160,115,160,125,157,125
1460 DATA157,115,-1,-1
1470 DATA240,70,242,58,252,60,250,72
1480 DATA240,70,-1,-1
1490 DATA275,43,276,32,270,31,272,24,285,5
1500 DATA286,15,275,43,-1,-1
1510 DATA160,290,135,290,124,302,95,310
1520 DATA85,306;95,300,75,302,85,295
1530 DATA75,293,87,285,75,285,85,280
1540 DATA85,275,100,270,85,270,100,245
1550 DATA110,245,110,240,95,238,85,242
1560 DATA98,228,90,228,90,218,120,216
1570 DATA130,210,118,202,130,190,150,185
1580 DATA175,190,185,217,165,230,170,265
1590 DATA155,285,160,290,-1,-1
```



You cannot be serious!!



MICRO CHOICE
YOUR BEST MICRO BUYER'S GUIDE
ADD-ONS AND PERIPHERALS FOR THE BBC MICRO AND ELECTRON
—THE ULTIMATE HARDWARE GUIDE
Which printers are worth writing about?
Advice and hints on how to choose your micro
Is extra memory worth the extra cost?
Are disc drives breaking any records?

About software, that is! Or at least you can't until you've made an intelligent decision about hardware.

Computers? Printers? Disc drives? Joysticks? Peripherals? Extra RAM?

I mean, there's a serious danger of going crazy just trying to understand the choices. Let alone coming to an intelligent decision.

Micro Choice is your answer. Every quarter it collects a range of hardware reports so that you can make your own choice of micro or add-ons.

Then you can forget about being too serious and start having fun. Easy when you think about it, isn't it?

MICRO CHOICE

DAVID REES

3D in Extended BASIC

Enlarge Extended BASIC to draw at speed and in 3D.

The human eye is probably the most powerful input device to the human mind, with a stereoscopic resolution of about 1500x1500 for colour and intensity. Thus, logically, the best way to get information is through pictures and, until a couple of decades ago, the best way to present viewed data was in 2D pictures.

However, with the advent of fast computers with hi-res graphics, full 3D representation and motion of objects became more feasible — perhaps the first well-known use of this was on the **2001** aircraft displays.

But this took a lot of hard work and expensive software. Now with these programs you'll be able to make and view 3D wire frame objects. A CBM64 plus my Extended BASIC (see elsewhere in this issue) will be able to plot them quickly — in fact, as twice as fast as the BBC micro.

3D DRAWING

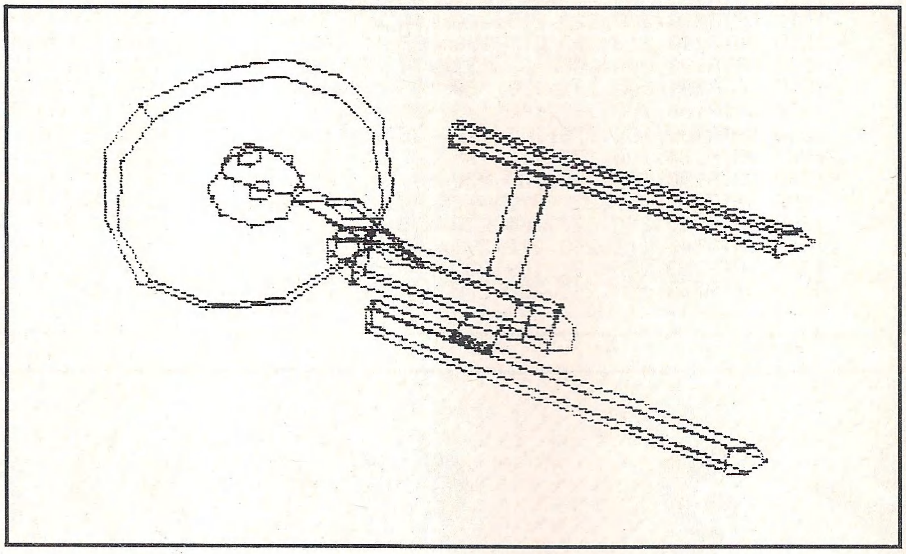
To accurately draw a 2D object, a graph with X and Y axes is used; to convert to 3D we need to add a Z axis. With these set, you can draw one view of a 3D object but see all angles you will need to rotate it.

Listing 1 gives you the basis of the 3D equations needed to do this The addition of a perspective equation gives the full 3D effect.

Listing 2 draws 3D objects on a 64 with Extended BASIC, using the DATA statements at the end of the program. The equations have been set out in the most streamlined and neat way and other speed increase additions make this a very fast BASIC routine.

However, even for small drawings, completion takes time so I have not included movement by key presses. Instead you can change variable numbers at the start of the program by reference to the variable list.

DATA in Listing 2 draws a simple house. However, if you want something a little more impressive, you can type in listing 3 which will give you a view of the **2001** space station instead.



CREATING SHAPES

If you want to create you own shapes, here are the DATA rules. DATA is set out as X,Y and Z coordinates, in the range of −63.99 to 63.99. 64,0,0 marks the end of a set of lines and the start of a new set. The end of all DATA is marked by 65.

VARIABLES USED

PH,TH,PS	*X,Y and Z rotation angles*
A1-A6	*resulting SIN and COS values*
PX,PY,PZ	*perspective: X moves Xwards, Y Ywards and Z changes magnification*
X,Y,Z	*actual 3D coordinates*
X%,Y%	*resulting screen coordinates*

FASTER STILL

The next listing, number 4, allows angles and perspectives to be changed by key entry. However, it is given as a guide and is not meant to be typed in.

Listing 5 POKEs custom machine code to memory and again needs Extended BASIC to function. After Listing 5, Listing 6 POKEs X,Y and Z positions to memory.

These can be in the range of −63.99 to 63.99 and, unlike most machine code routines, non whole numbers can be used. The accuracy can be thought of as being + or − 16383 divided by 256.

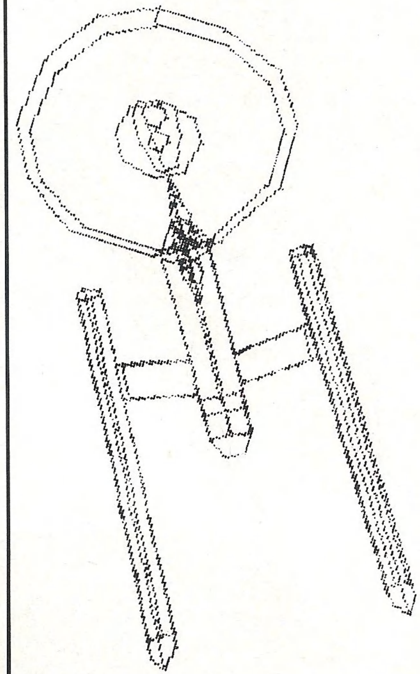
64,0,0 signifies the end of a set of 4 lines and 65,0,0 the end of DATA. For those wishing to create their own objects, use the above information to replace the DATA with your own.

MOVING

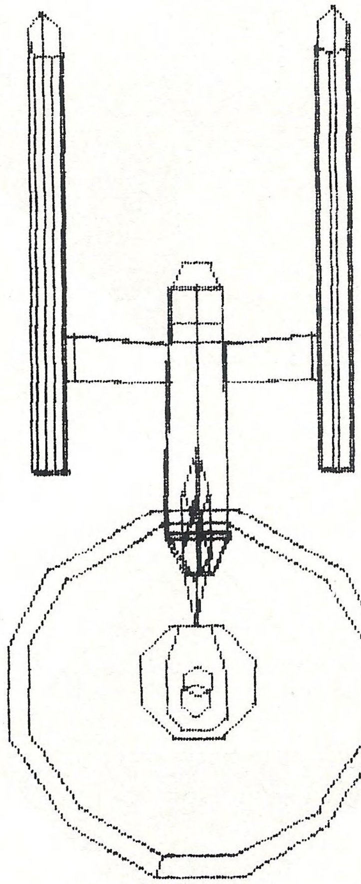
Listing 7 utilises the routine to make it easier for you to create and move the picture. Variables in lines 80-90 can be changed to most values (range as with X,Y and Z) to change the start position of your object — the table shows key functions. Note that you can only print out a screen once; to repeat you will have to restart the program.

By changing POKE 31555 in line 30 you can change the graphics colour. Or, by changing POKE 31554 to zero, you can draw points instead of lines. Or experiment with the commands in Extended BASIC.

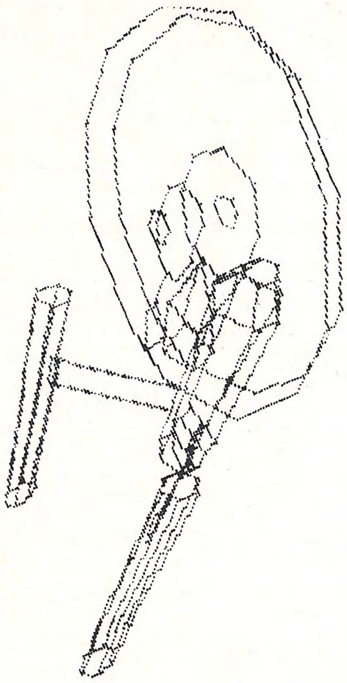
```
90 REM*   3D EQUATION    *
91 REM*A1 TO A6 REPRESENT*
92 REM*SIN AND COS VALUES*
93 REM*  PRE-CALCULATED  *
100 Y1=Y*A2-Z*A1:Z=Y*A1+Z*A2
120 X1=X*A4-Z*A3:Z=X*A3+Z*A4
140 X=X1*A6-Y1*A5:Y=X1*A5+Y1*A6
160 X=2*X+PX:Y=2*Y+PY:Z=Z+PZ
170 Z=(256-Z)*(256-Z)/256
178 REM*128 AND 99 ARE NUMBERS*
179 REM* FOR SCREEN CENTERING *
180 X=128-X*Z/256:Y=99-Y*Z/256
```

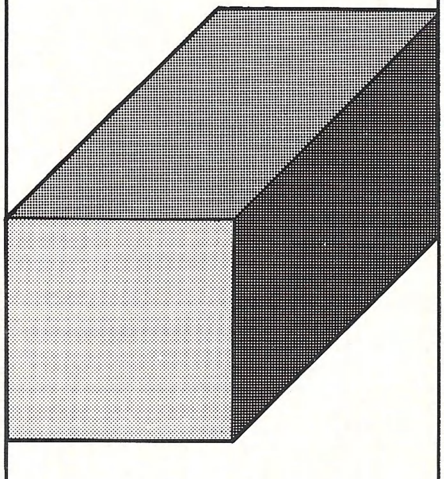


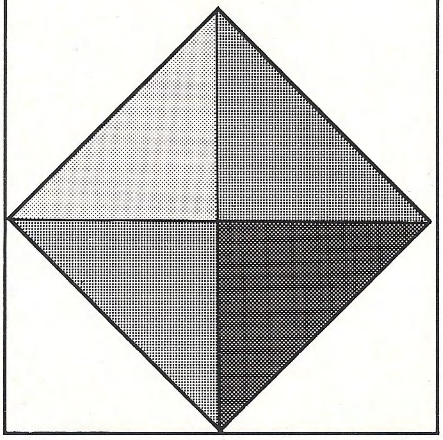
```
1 REM*FAST BASIC 3D ROUTINE*
2 REM*    BY DAVID REES     *
5 PH=-π/3:TH=π/16:PS=0
6 A1=SIN(PH):A2=COS(PH):A3=SIN(TH)
7 A4=COS(TH):A5=SIN(PS):A6=COS(PS)
8 POKE56325,255
10 MODE=51828:GCOL=51770:GMOVE=52039
20 LINE=51754:GTEXT=52657
30 SYSGCOL,5,0:SYSGCOL,6,0
40 SYSMODE,0,0,1:SYSGCOL,1,6
50 PX=-50:PY=0:FZ=-63
60 FORN=1TO31
70 READX,Y,Z:IFX=64THEN60
80 IFX=65THEN2000
100 Y1=Y*A2-Z*A1:Z=Y*A1+Z*A2
120 X1=X*A4-Z*A3:Z=X*A3+Z*A4
140 X=X1*A6-Y1*A5:Y=X1*A5+Y1*A6
160 X=2*X+PX:Y=2*Y+PY:Z=Z+PZ
170 Z=(256-Z)*(256-Z)/256
180 X=128-X*Z/256:Y=99-Y*Z/256
200 X%=X*(X>0)*(X<320)-319*(X>319)
210 Y%=Y*(Y>0)*(Y<200)-199*(Y>199)
260 IFN=1THENSYSGMOVE,X%,Y%:GOTO300
270 SYSLINE,2,X%,Y%
300 NEXT
1000 DATA-30,20,-20,-30,-20,-20,-30,-20,
1010 DATA-30,20,20,-30,20,-20,30,20,-20
1020 DATA30,20,20,-30,20,20,-30,,30,-30
1030 DATA-20,20,30,-20,20,30,,30,30,20,2
1040 DATA30,-20,20,30,-20,-20,-30,-20,-2
1050 DATA64,,,-30,,30,30,,30,64,,
1060 DATA30,20,-20,30,-20,-20,64,,
1070 DATA30,10,-20,30,-10,-20,30,-10,40
1080 DATA30,10,40,30,10,-20,40,10,-20
1090 DATA40,10,40,40,-10,40,40,-10,-20
1100 DATA40,10,-20,64,,
1110 DATA30,10,40,40,10,40,64,,
1120 DATA30,-10,40,40,-10,40,64,,
1130 DATA30,-10,-20,40,-10,-20,65,,
2000 GETA$:IFA$=""THEN2000
2010 SYSMODE,6,0,0
2020 PRINT"⌐"
2030 POKE56325,68
```



▷





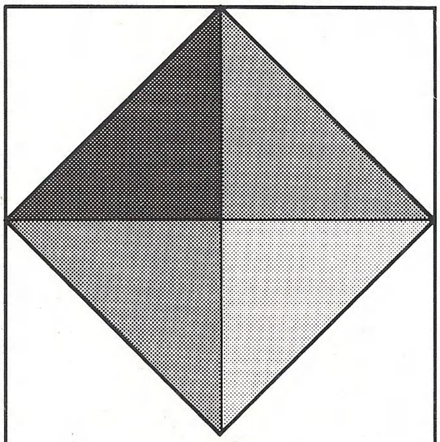


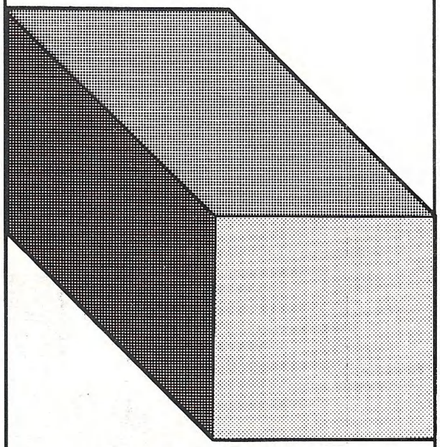
```
10 REM********************
11 REM*   DATA FOR THE    *
12 REM*2001 SPACE STATION*
13 REM*   FOR BOTH THE    *
14 REM*BASIC  AND MACHINE*
15 REM* CODE 3D PROGRAMS *
16 REM*        BY         *
17 REM*    DAVID REES     *
18 REM********************
1000 DATA-44,12,11,-31,32,11,-12,44,11,12,44,11
1010 DATA31,32,11,44,12,11,44,-12,11,31,-32,11
1020 DATA12,-44,11,-12,-44,11,-31,-32,11
1030 DATA-44,-12,11,-44,12,11,-37,10,11
1040 DATA-27,28,11,-9,38,11,9,38,11,27,28,11
1050 DATA37,10,11,37,-10,11,27,-28,11,9,-38,11
1060 DATA-9,-38,11,-27,-28,11,-37,-10,11
1065 DATA-37,10,11
1070 DATA-37,10,7,-27,28,7,-9,38,7,9,38,7
1080 DATA27,28,7,37,10,7,37,-10,7,27,-28,7
1090 DATA9,-38,7,-9,-38,7,-27,-28,7,-37,-10,7
1100 DATA-37,10,7,-44,12,7,-31,32,7,-12,44,7
1110 DATA12,44,7,31,32,7,44,12,7,44,-12,7
1120 DATA31,-32,7,12,-44,7,-12,-44,7,-31,-32,7
1130 DATA-44,-12,7,-44,12,7,64,,
1140 DATA-7,-2.5,7,-37,-2.5,7,-37,-2.5,11
1150 DATA-7,-2.5,11,-7,2.5,11
1160 DATA-37,2.5,11,-37,2.5,7,-7,2.5,7
1170 DATA-2.5,7,7,-2.5,38,7,-2.5,38,11
1180 DATA-2.5,7,11,2.5,7,11,2.5,38,11
1190 DATA2.5,38,7,2.5,7,7,7,2.5,7
1200 DATA37,2.5,7,37,2.5,11,7,2.5,11
1210 DATA7,-2.5,11,37,-2.5,11,37,-2.5,7
1220 DATA7,-2.5,7,2.5,-7,7,2.5,-37,7
1230 DATA2.5,-37,11,2.5,-7,11,-2.5,-7,11
1240 DATA-2.5,-37,11,-2.5,-37,7,-2.5,-7,7
1250 DATA-7,-2.5,7,-7,-2.5,11,-7,2.5,11
1260 DATA-2.5,7,11,2.5,7,11,7,2.5,11
1270 DATA7,-2.5,11,2.5,-7,11,-2.5,-7,11
1280 DATA-7,-2.5,11,64,,
1290 DATA-5,2,11,5,2,11,5,-2,11
1300 DATA-5,-2,11,-5,2,11,64,,
1310 DATA-2.5,-2.5,7,-2.5,-2.5,-7,-2.5,2.5,-7
1320 DATA-2.5,2.5,7,2.5,2.5,7,2.5,2.5,-7
1330 DATA2.5,-2.5,-7,2.5,-2.5,7,-2.5,-2.5,7
1340 DATA64,,
1400 DATA-44,12,-11,-31,32,-11,-12,44,-11,12,44
1410 DATA-11,31,32,-11,44,12,-11,44,-12,-11,31
1420 DATA-32,-11,12,-44,-11,-12,-44,-11,-31;-32
1430 DATA-11,-44,-12,-11,-44,12,-11,-37,10,-11
1440 DATA-27,28,-11,-9,38,-11,9,38,-11,27,28,-11
1450 DATA37,10,-11,37,-10,-11,27,-28,-11,9,-38
1460 DATA-11,-9,-38,-11,-27,-28,-11,-37,-10,-11
1465 DATA-37,10,-11,-37
1470 DATA10,-7,-27,28,-7,-9,38,-7,9,38,-7,27,28
1480 DATA-7,37,10,-7,37,-10,-7,27,-28,-7,9,-38
1490 DATA-7,-9,-38,-7,-27,-28,-7,-37,-10,-7,-37
1500 DATA10,-7,-44,12,-7,-31,32,-7,-12,44,-7,12
1510 DATA44,-7,31,32,-7,44,12,-7,44,-12,-7,31
1520 DATA-32,-7,12,-44,-7,-12,-44,-7,-31,-32,-7
1530 DATA-44,-12,-7,-44,12,-7,64,,
1540 DATA-7,-2.5,-7,-37,-2.5,-7,-37,-2.5,-11
1550 DATA-7,-2.5,-11,-7,2.5,-11
1560 DATA-37,2.5,-11,-37,2.5,-7,-7,2.5,-7
1570 DATA-2.5,7,-7,-2.5,38,-7,-2.5,38,-11
```

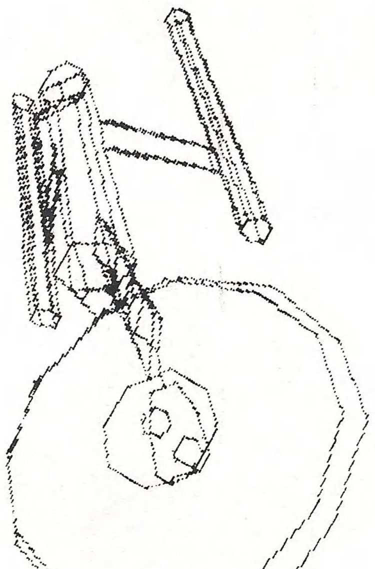
```
1580 DATA-2.5,7,-11,2.5,7,-11,2.5,38,-11
1590 DATA2.5,38,-7,2.5,7,-7,7,2.5,-7
1600 DATA37,2.5,-7,37,2.5,-11,7,2.5,-11
1610 DATA7,-2.5,-11,37,-2.5,-11,37,-2.5,-7
1620 DATA7,-2.5,-7,2.5,-7,-7,2.5,-37,-7
1630 DATA2.5,-37,-11,2.5,-7,-11,-2.5,-7,-11
1640 DATA-2.5,-37,-11,-2.5,-37,-7,-2.5,-7,-7
1650 DATA-7,-2.5,-7,-7,-2.5,-11,-7,2.5,-11
1660 DATA-2.5,7,-11,2.5,7,-11,7,2.5,-11
1670 DATA7,-2.5,-11,2.5,-7,-11,-2.5,-7,-11
1680 DATA-7,-2.5,-11,65,,
```

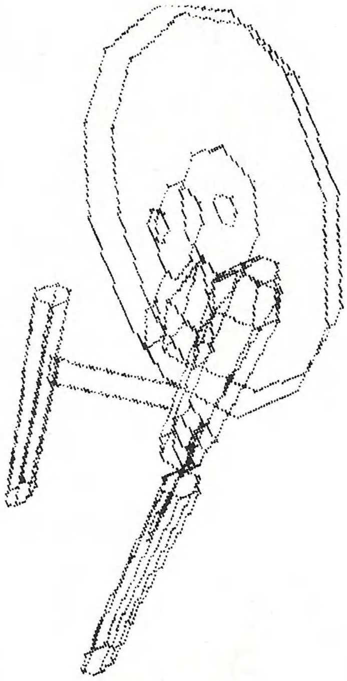
```
 90 REM*   3D EQUATION    *
 91 REM*A1 TO A6 REPRESENT*
 92 REM*SIN AND COS VALUES*
 93 REM*  PRE-CALCULATED  *
 100 Y1=Y*A2-Z*A1:Z=Y*A1+Z*A2
 120 X1=X*A4-Z*A3:Z=X*A3+Z*A4
 140 X=X1*A6-Y1*A5:Y=X1*A5+Y1*A6
 160 X=2*X+PX:Y=2*Y+PY:Z=Z+PZ
 170 Z=(256-Z)*(256-Z)/256
 178 REM*128 AND 99 ARE NUMBERS*
 179 REM* FOR SCREEN CENTERING *
 180 X=128-X*Z/256:Y=99-Y*Z/256
```

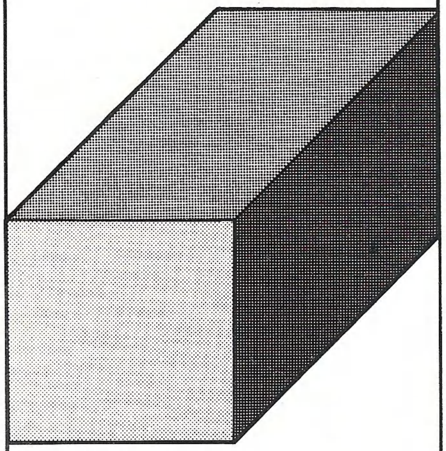
```
1 REM*MACHINE CODE FOR*
2 REM*  FAST 3D DRAW  *
3 REM* BY  DAVID REES *
4 REM*   FEB  1984    *
5 POKE52,63:POKE56,63
10 X=37376
110 FORN=0TO51
120 FORM=0TO955
130 READA:C=C+A:IFA=-1THEN150
140 POKEX+M+64*N,A:NEXT
150 NEXTN
160 PRINTC,"IT SHOULD BE 187871"
1000 DATA169,,141,64,123,141,65,123
1010 DATA172,64,123,185,,114,201,2
1020 DATA208,14,169,,141,65,123,238,64,123
1030 DATA173,64,123,208,233,96
1035 DATA201,3,208,1,96,141,27,123
1040 DATA185,,115,141,29,123,185,,116,141,30,123
1050 DATA185,,117,141,31,123,185,,118,141,33,123
1060 DATA185,,119,141,34,123,185,,120,141,35,123
1070 DATA185,,121,141,37,123,185,,122,141,38,123
1080 DATA169,,141,28,123,141,32,123,141,36,123
1090 DATA32,,148,173,66,123,208,24
1100 DATA173,28,123,141,92,3,173,29,123
1110 DATA141,93,3,173,33,123,141,94,3
1120 DATA32,105,194,24,144,77,173,63,3
1130 DATA141,60,3,173,64,3,141,61,3
1140 DATA173,65,3,141,62,3,173,28,123
1150 DATA141,63,3,173,29,123,141,64,3
1160 DATA173,33,123,141,65,3,173,67,123
1170 DATA141,89,3,173,60,3,205,63,3
1172 DATA208,19,173,61,3,205,64,3,208,11
1174 DATA173,62,3,205,65,3,208,3,24,144,8
1178 DATA173,65,123,240,3,32,112,197
1180 DATA238,65,123,238,64,123,173,64,123
1190 DATA208,1,96,76,8,146,-1,-1,-1,-1
1200 DATA185,,123,133,80,185,1,123,133,81
```

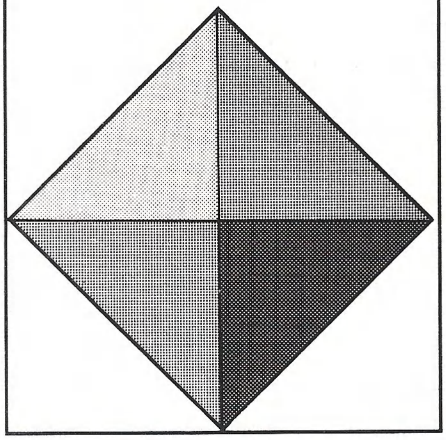






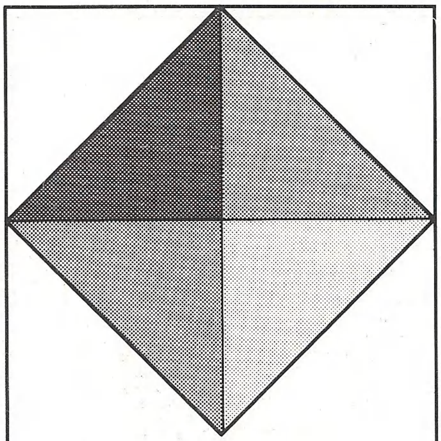


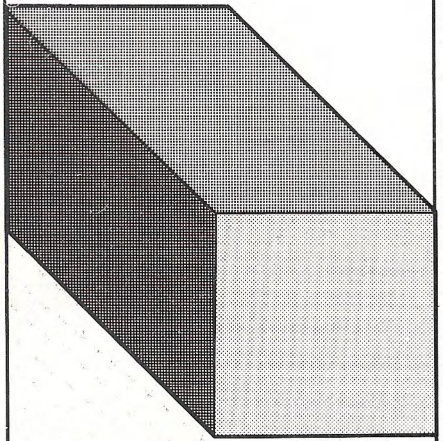


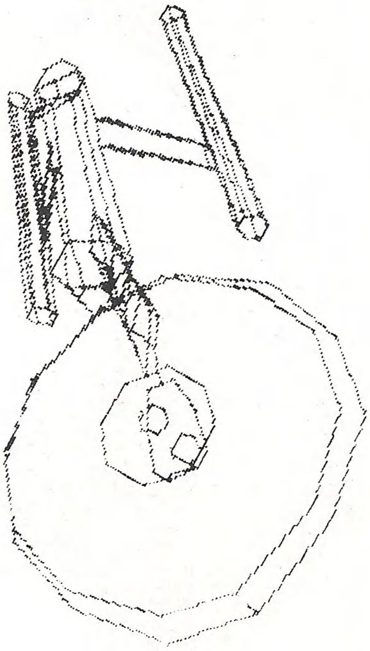


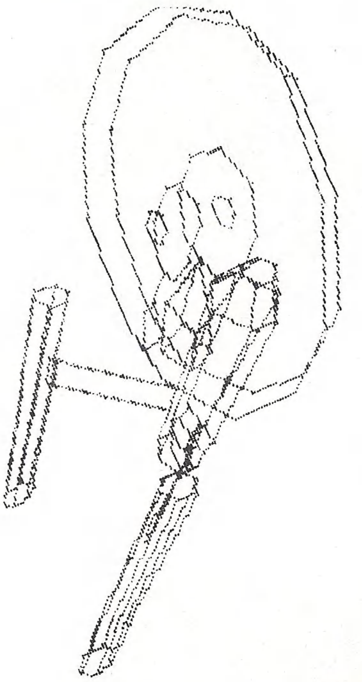
```
1210 DATA185,2,123,133,82,185,3,123,133,83
1220 DATA172,56,123,185,,123,133,84
1230 DATA185,1,123,133,86,185,2,123
1240 DATA133,87,169,,133,85,32,,157
1250 DATA165,88,141,47,123,165,89,141,48,123
1260 DATA165,90,141,49,123,165,91,141,50,123
1270 DATA172,57,123,185,,123,133,80
1280 DATA185,1,123,133,81,185,2,123,133,82
1290 DATA185,3,123,133,83,172,58,123
1300 DATA185,,123,133,84,185,1,123,133,86
1310 DATA185,2,123,133,87,169,,133,85
1320 DATA32,,157,165,88,133,80,165,89,133,81
1330 DATA165,90,133,82,165,91,133,83
1340 DATA173,47,123,133,84,173,48,123,133,85
1350 DATA173,49,123,133,86,173,50,123,133,87
1360 DATA173,60,123,240,6,32,128,158,24
1370 DATA144,3,32,,158,172,59,123,165,88
1380 DATA153,,123,165,89,153,1,123,165,90
1390 DATA153,2,123,165,91,153,3,123,96
1400 DATA-1,-1,-1,-1
1410 DATA169,1,141,60,123,169,12,141,56,123
1420 DATA169,35,141,57,123,169,9,141,58,123
1430 DATA169,43,141,59,123,160,31,32,,147
1440 DATA169,,141,60,123,169,9,141,56,123
1450 DATA169,35,141,57,123,169,12,141,58,123
1460 DATA169,35,141,59,123,160,31,32,,147
1470 DATA169,1,141,60,123,169,18,141,56,123
1480 DATA169,35,141,57,123,169,15,141,58,123
1490 DATA169,39,141,59,123,160,27,32,,147
1500 DATA169,,141,60,123,169,15,141,56,123
1510 DATA169,35,141,57,123,169,18,141,58,123
1520 DATA169,35,141,59,123,160,27,32,,147
1530 DATA169,1,141,60,123,169,24,141,56,123
1540 DATA169,43,141,57,123,169,21,141,58,123
1550 DATA169,27,141,59,123,160,39,32,,147
1560 DATA169,,141,60,123,169,21,141,56,123
1570 DATA169,43,141,57,123,169,24,141,58,123
1580 DATA169,31,141,59,123,160,39,32,,147
1590 DATA173,27,123,133,80,173,28,123,133,81
1600 DATA173,29,123,133,82,173,30,123,133,83
1610 DATA24,38,83,38,82,38,81,24
1620 DATA173,,123,133,84,169,,133,85
1630 DATA173,1,123,133,86,173,2,123,133,87
1640 DATA32,,158,165,88,141,27,123,165,89
1650 DATA141,28,123,165,90,141,29,123
1660 DATA165,91,141,30,123,173,31,123
1670 DATA133,80,173,32,123,133,81,173,33,123
1680 DATA133,82,173,34,123,133,83
1690 DATA24,38,83,38,82,38,81,24
1700 DATA173,3,123,133,84,169,,133,85
1710 DATA173,4,123,133,86,173,5,123,133,87
1720 DATA32,,158,165,88,141,31,123
1730 DATA165,89,141,32,123,165,90,141,33,123
1740 DATA165,91,141,34,123,173,35,123
1745 DATA133,80,173,36,123,133,81,173,37,123
1750 DATA133,82,173,38,123,133,83,173,6,123
1755 DATA133,84,169,,133,85,173,7,123
1760 DATA133,86,173,8,123,133,87,32,,158
1765 DATA165,88,133,80,165,89,133,81
1770 DATA165,90,133,82,165,91,133,83
1775 DATA169,,133,84,133,85,133,87,169,128
1776 DATA133,86,24,102,81,102,82,102,83,24
1780 DATA32,128,158,165,88,133,80
1785 DATA133,84,165,89,133,81,133,85
1790 DATA165,90,133,82,133,86,165,91
```

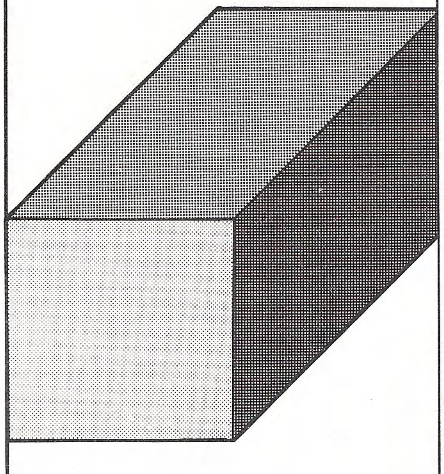
```
1795 DATA133,83,133,87
1797 DATA24,38,87,38,86,38,85,24
1799 DATA32,,156,165,88
1800 DATA141,35,123,169,,141,36,123
1805 DATA165,89,141,37,123,165,90,141,38,123
1810 DATA173,27,123,133,80,173,28,123,133,81
1815 DATA173,29,123,133,82,173,30,123,133,83
1820 DATA173,35,123,133,84,173,36,123,133,85
1825 DATA173,37,123,133,86,173,38,123,133,87
1830 DATA32,,156,165,88,133,80,169,,133,84
1835 DATA133,85,133,87,133,81,165,89
1840 DATA133,82,165,90,133,83,169,128,133,86
1845 DATA32,128,158,24,38,91,38,90,38,89,24
1846 DATA165,90,56,237,98,123,176,17
1847 DATA133,90,198,89,165,89,201,255,208,10
1848 DATA169,1,133,88,24,144,3,24,133,90
1850 DATA165,88,240,17,169,,141,27,123
1855 DATA141,28,123,141,29,123,141,30,123
1860 DATA24,144,54,141,27,123,165,89
1865 DATA201,,240,32,201,1,208,7
1868 DATA165,90,56,233,64,144,19
1870 DATA24,169,63,141,29,123,169,1
1875 DATA141,28,123,169,,141,30,123,24
1880 DATA144,15,165,89,141,28,123,165,90
1881 DATA141,29,123,165,91,141,30,123
1885 DATA173,31,123,133,80,173,32,123,133,81
1890 DATA173,33,123,133,82,173,34,123,133,83
1895 DATA173,35,123,133,84,173,36,123,133,85
1900 DATA173,37,123,133,86,173,38,123,133,87
1905 DATA32,,156,165,88,133,80,169,,133,84
1910 DATA133,85,133,87,133,81,165,89,133,82
1915 DATA165,90,133,83,169,87,133,86
1920 DATA32,128,158,24,38,91,38,90,38,89,24
1921 DATA165,90,56,237,99,123,176,17
1922 DATA133,90,198,89,165,89,201,255,208,10
1923 DATA169,1,133,88,24,144,3,24,133,90
1925 DATA165,88,240,15,169,,141,31,123
1930 DATA141,32,123,141,33,123,141,34,123
1935 DATA96,141,31,123,165,89,240,2
1940 DATA169,
1945 DATA141,32,123,165,90,56,233,200
1950 DATA144,6,24,169,199,24,144,2
1960 DATA165,90,141,33,123,165,91,141,34,123
1965 DATA96
1990 DATA-1,-1
1991 DATA-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1992 DATA-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1993 DATA-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
2000 DATA165,81,141,199,3,169,1,133,38
2010 DATA160,8,169,,133,89,133,90
2020 DATA133,91,165,87,37,38,201,
2030 DATA240,29,165,90,24,101,81,144,3
2040 DATA24,230,89,133,90,165,91,24
2050 DATA101,82,144,8,24,230,90,208,3
2060 DATA24,230,89,133,91,24,6,38,24
2070 DATA38,83,38,82,38,81,24,136
2080 DATA208,205,165,82,133,83,165,81
2090 DATA133,82,173,199,3,133,81
2100 DATA 169,1,133,38,160,8,165,86
2110 DATA37,38,201,,240,37,165,89,24
2120 DATA101,81,24,133,89,165,90,24
2130 DATA101,82,144,3,24,230,89,133,90
2140 DATA165,91,24,101,83,144,8,24
2150 DATA230,90,208,3,24,230,89,133,91
2160 DATA24,6,38,24,38,83,38,82,38,81
```

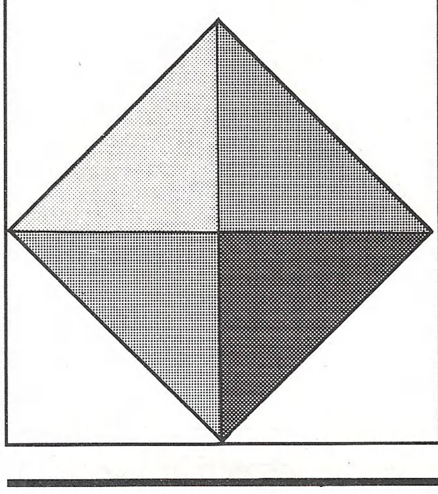






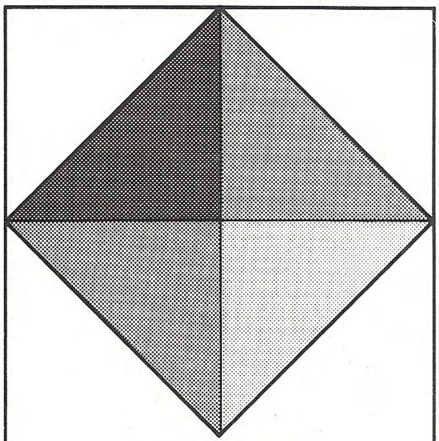


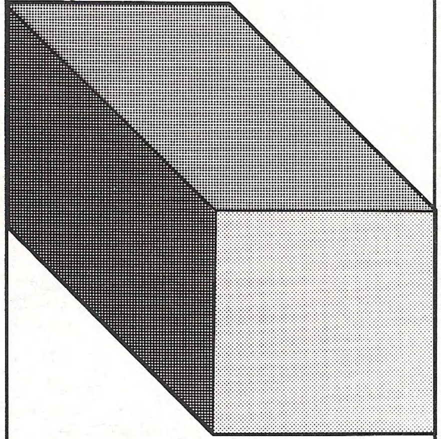


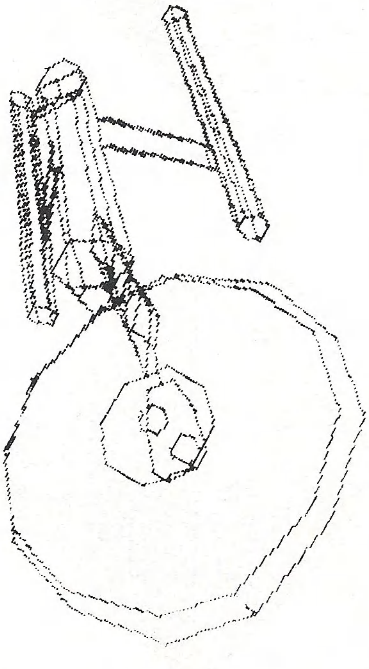


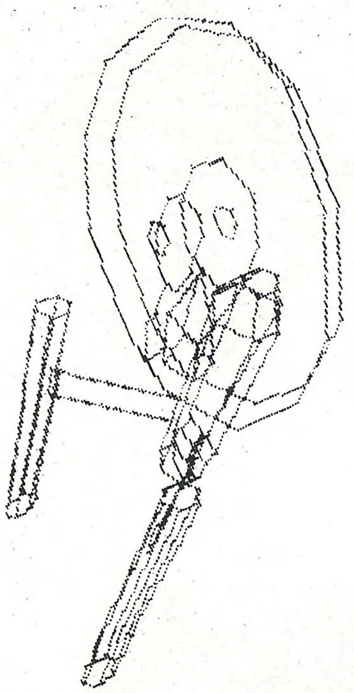
```
2170 DATA24,136,208,197
2180 DATA165,85,41,1,240,20,165,89,24
2190 DATA101,81,24,133,89,165,90,24
2200 DATA101,82,144,3,24,230,89
2210 DATA133,90,165,80,24,101,84,24
2220 DATA41,1,133,88,96,-1,-1,-1
2370 DATA160,,185,80,,153,,124,200
2380 DATA192,12,208,245,160,,185,,125
2390 DATA153,80,,200,192,12,208,245
2400 DATA32,,146,160,,185,80,,153,,126
2410 DATA200,192,12,208,245,160,
2420 DATA185,,124,153,80,,200,192,12
2430 DATA208,245,96,-1
2440 DATA165,81,141,199,3,169,1,133,88
2450 DATA160,8,169,,133,89,133,90
2460 DATA133,91,165,87,37,38,201,
2470 DATA240,29,165,90,24,101,81,144,3
2480 DATA24,230,89,133,90,165,91,24
2490 DATA101,82,144,8,24,230,90,208,3
2500 DATA24,230,89,133,91,24,6,38,24
2510 DATA38,83,38,82,38,81,24,136
2520 DATA208,205,165,82,133,83,165,81
2530 DATA133,82,173,199,3,133,81
2540 DATA165,86
2550 DATA41,1,201,,240,37,165,89,24
2560 DATA101,81,24,133,89,165,90,24
2570 DATA101,82,144,3,24,230,89,133,90
2580 DATA165,91,24,101,83,144,8,24
2590 DATA230,90,208,3,24,230,89,133,91
2600 DATA24,165,80,24,101,84,24
2610 DATA41,1,133,88,96,-1,-1,-1
2620 DATA164,80,165,84,133,80,132,84
2630 DATA164,81,165,85,133,81,132,85
2640 DATA164,82,165,86,133,82,132,86
2650 DATA164,83,165,87,133,83,132,87
2660 DATA96,-1
2670 DATA165,80,240,11,165,84,208,21
2680 DATA169,,133,80,76,164,156
2690 DATA165,84,240,10,32,192,157
2700 DATA169,,133,80,76,164,158
2710 DATA165,81,24,101,85,24,133,89
2720 DATA165,82,24,101,86,144,3,24
2730 DATA230,89,133,90,165,83,24,101,87
2740 DATA144,8,24,230,90,208,3,24
2750 DATA230,89,133,91,165,80,133,88
2760 DATA96,-1,-1
2770 DATA165,80,240,21,169,,133,80
2780 DATA165,84,240,10,32,192,157
2790 DATA169,,133,80,24,144,14,76,29,158
2800 DATA165,84,240,7,169,1,133,80
2810 DATA76,29,158,169,,133,88,165,85
2820 DATA56,229,81,176,9,133,89
2830 DATA169,1,133,88,24,144,3,24
2840 DATA133,89,165,86,56,229,82
2850 DATA176,12,198,89,166,89
2860 DATA224,255,208,4,160,1,132,88
2880 DATA24,133,90 ,165,87,56,229,83
2890 DATA176,21,198,90
2900 DATA166,90,224,255,208,13,24,198,89
2910 DATA166,89,224,255,208,4
2920 DATA160,1,132,88,24,133,91
2940 DATA 165,88,208,1,96
2950 DATA165,89,73,255,133,89,165,90
2960 DATA73,255,133,90,165,91,73,255
2970 DATA133,91,96,-1,-1
```

```
5 REM*GRAPHIC DATA FOR THE*
6 REM*   USS ENTERPRISE   *
7 REM*       NCC 1701     *
10 S=29184:FORN=0TO255:POKES+N,3:NEXT
20 T=256:IFPEEK(49152)<>173THENEND
30 FORN=0TO255
40 READX,Y,Z:IFX=64THEN200
50 IFX=65THEN300
60 A=-SGN(X):X=ABS(X):IFA=-1THENA=0
70 B=INT(X):D=(X-B)*256:C=BAND255:B=INT(B/256)
80 POKES+N,A:POKES+T+N,C:POKES+T*2+N,D
100 A=-SGN(Y):Y=ABS(Y):IFA=-1THENA=0
110 B=INT(Y):D=(Y-B)*256:C=BAND255:B=INT(B/256)
120 POKES+T*3+N,A:POKES+T*4+N,C:POKES+T*5+N,D
140 A=-SGN(Z):Z=ABS(Z):IFA=-1THENA=0
150 B=INT(Z):D=(Z-B)*256:C=BAND255:B=INT(B/256)
160 POKES+T*6+N,A:POKES+T*7+N,C:POKES+T*8+N,D
180 NEXTN
190 END
200 POKES+N,2:GOTO180
300 POKES+N,3:GOTO190
320 POKES+N,3:GOTO190
1000 DATA-60,6,9,-52,18,9,-40,26,9
1010 DATA-28,26,9,-16,20,9,-8,6,9
1020 DATA-8,-6,9,-16,-20,9,-28,-26,9
1030 DATA-40,-26,9,-52,-18,9,-60,-6,9
1040 DATA-60,6,9,-58,5,6,-51,17,6
1050 DATA-40,24,6,-28,24,6,-18,18,6
1060 DATA-10,5,6,-10,-5,6,-18,-18,6
1070 DATA-28,-24,6,-40,-24,6,-51,-17,6
1080 DATA-58,-5,6,-58,5,6,64,,
1090 DATA-24,,10,-24,-2,10,-30,-4,11
1100 DATA-36,-4,11,-38,-2,11,-38,2,11
1110 DATA-36,4,11,-30,4,11,-24,2,10
1120 DATA-24,,10,-18,,10,-14,2,9
1130 DATA-8,,9,-14,-2,9,-18,,10,64,,
1140 DATA-34,-2,14,-32,-2,14,-30.7,,14
1150 DATA-32,2,14,-34,2,14,-35.3,,14
1160 DATA-34,-2,14,64,,
1170 DATA-25,,5,-25,3.5,5,-30,8,5
1180 DATA-36,8,5,-41,3.5,5,-41,-3.5,5
1190 DATA-36,-8,5,-30,-8,5,-25,-3.5,5
1200 DATA-25,,5,-12,,-1,-6,-2,-1.8
1210 DATA-15,-2,6,-25,,5,-15,2,6
1220 DATA-6,2,-1.8,1,,-2,-7,,6
1230 DATA-15,-2,6,64,,
1240 DATA-15,2,6,-7,,6,64,,
1250 DATA-12,,-1,-6,2,-1.8,64,,
1260 DATA-6,-2,-1.8,1,,-2,64,,
1270 DATA-19,,-10,-13,,-13,18,,-10
1280 DATA21,,-8,26,,-8,26,-4,-8
1290 DATA21,-4,-8,18,-4.25,-9,-13,-5,-9
1300 DATA-19,-2,-8,-19,-2,-5,-13,-5,-4.5
1310 DATA26,-4,-5,26,,-3,-13,,-1
1320 DATA-19,,-4,-19,2,-5,-13,5,-4.5
1330 DATA26,4,-5,26,4,-8,30,2,-8
1340 DATA30,-2,-8,26,-4,-8,26,4,-8
1350 DATA21,4,-8,18,4.25,-9,-13,5,-9
1360 DATA-19,2,-8,-19,,-10,64,,
1370 DATA26,4,-5,26,,-3,64,,
1380 DATA26,-4,-5,26,-4,-8,64,,
1390 DATA21,-4,-8,21,4,-8,64,,
1400 DATA18,-4.25,-9,18,4.25,-9,64,,
1410 DATA-13,,-13,-13,-5,-9,-13,-5,-4.5
1420 DATA-13,,-1,-13,5,-4.5,-13,5,-9
```

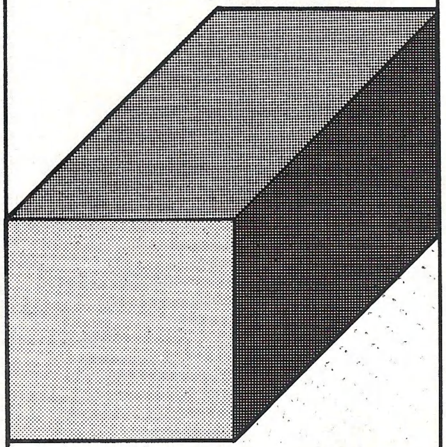






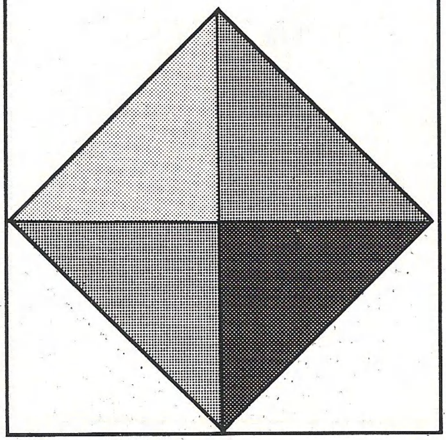


```
1430 DATA-13,,-13,64,,
1440 DATA-19,,-10,-19,-2,-8,64,,
1450 DATA-19,-2,-5,-19,,-4,64,,
1460 DATA-19,2,-5,-19,2,-8,64,,
1470 DATA-34,-2,4,-32,-2,4,-30.7,,4
1480 DATA-32,2,4,-34,2,4,-35.3,,4
1490 DATA-34,-2,4,64,,
1500 DATA11,16,11,11,4,-4,11,5,-3
1510 DATA11,17,12,17,17,12,17,5,-3
1520 DATA17,4,-4,17,16,11,11,16,11
1530 DATA64,,
1540 DATA-2,19,10,56,19,10,56,17,11.5
1550 DATA-2,17,11.5,-2,17,14.5
1560 DATA58,17,14.5,60,19,16,-2,19,16
1570 DATA-2,21,14.5,58,21,14.5
1580 DATA56,21,11.5,-2,21,11.5
1590 DATA-2,19,10,64,,
1600 DATA56,17,11.5,58,17,14.5,64,,
1610 DATA60,19,16,58,21,14.5,64,,
1620 DATA56,21,11.5,56,19,10,64,,
1630 DATA-2,17,11.5,-2,19,10,64,,
1640 DATA-2,17,14.5,-2,19,16,64,,
1650 DATA-2,21,11.5,-2,21,14.5,64,,
1700 DATA11,-16,11,11,-4,-4,11,-5,-3
1710 DATA11,-17,12,17,-17,12,17,-5,-3
1720 DATA17,-4,-4,17,-16,11,11,-16,11
1730 DATA64,,
1740 DATA-2,-19,10,56,-19,10,56,-17,11.5
1750 DATA-2,-17,11.5,-2,-17,14.5
1760 DATA58,-17,14.5,60,-19,16,-2,-19,16
1770 DATA-2,-21,14.5,58,-21,14.5
1780 DATA56,-21,11.5,-2,-21,11.5
1790 DATA-2,-19,10,64,,
1800 DATA56,-17,11.5,58,-17,14.5,64,,
1810 DATA60,-19,16,58,-21,14.5,64,,
1820 DATA56,-21,11.5,56,-19,10,64,,
1830 DATA-2,-17,11.5,-2,-19,10,64,,
1840 DATA-2,-17,14.5,-2,-19,16,64,,
1850 DATA-2,-21,11.5,-2,-21,14.5,65,,
```



KEY FUNCTIONS

Q/W	*changes X angle*
E/R	*changes Y angle*
T/Y	*changes Z angle*
P	*printout to VIC 1525 printer*
D	*draw the shape*
1/2	*change X perspective*
3/4	*change Y perspective*
5/6	*change Z perspective*



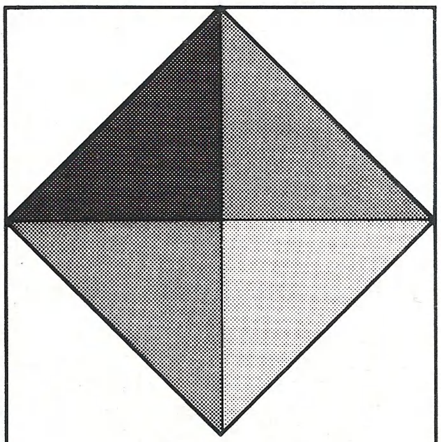
VARIABLES USED

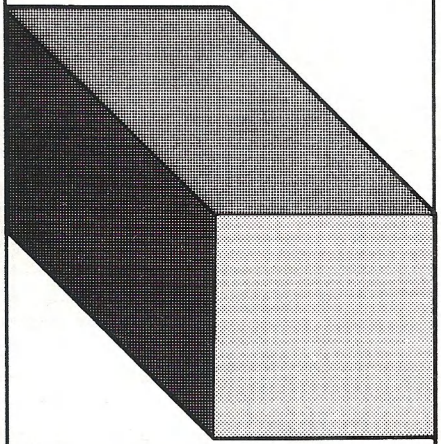
Q	*start of machine code registers*
M	*what is added/substracted from an angle with a key press*
B1-B3	*angles for X,Y and Z rotation*
A1-A6	*SIN and COS values for angles B1-B3*
PX,PY,PZ	*perspective: PX moves image Xwards, PY Ywards and PZ for magnification*

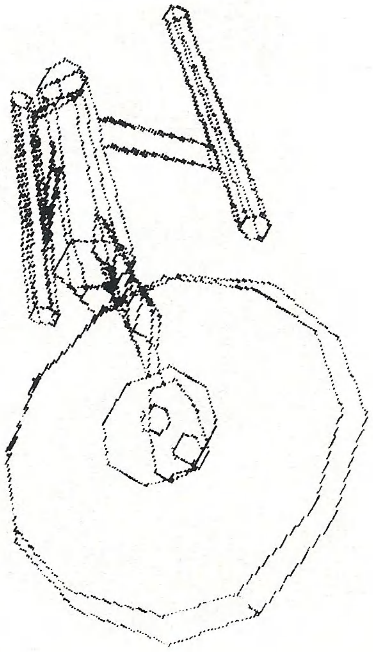
HOW IT RUNS

5	*set memory limits*
10-15	*set Extended BASIC commands*
20	*set screen colours*
30	*set line colour*
40	*set POINT or LINE drawing*
50	*set X and Y shift*
80	*set start angles*
85	*POKE start angles to memory*
90-95	*set and POKE perspective*
100-146	*GET a key and move to a routine*
150-220	*change angles routine*
500-520	*POKE new values routine*
1000-1030	*draw new image*
2000-2370	*make hi-res printout*
4000-4030	*spare a print routine*

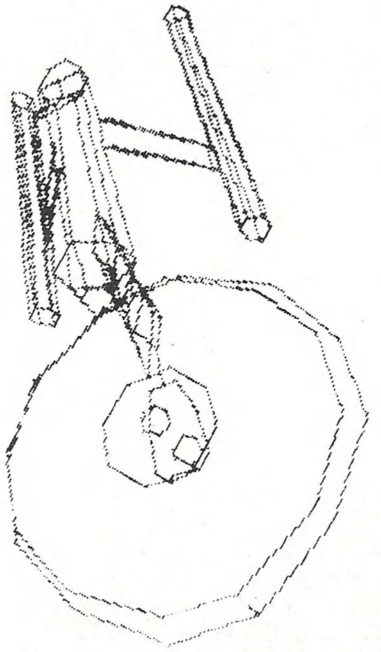
```
5 POKE51,0:POKE52,31:POKE55,0:POKE56,31
10 MODE=51828:GCOL=51770:GTEXT=52657
15 LINE=51754:GMOVE=52039
20 SYSGCOL,5,0:SYSGCOL,6,0:SYSGCOL,1,14
29 REM*LINE COLOUR*
30 POKE31555,14:POKE56325,255
39 REM*POINT(0) OR LINE(1)*
40 POKE31554,1:Q=123*256
45 REM*X,Y*
50 POKE31586,100:POKE31587,90
80 B1=π*2.07:B2=π*3.1591:B3=π*-3.083:M=π/12
85 GOSUB160:GOSUB190:GOSUB220
90 PX=0:PY=0:PZ=10:GOSUB240:GOSUB260
95 GOSUB280:V=40128
 100 GETA$:IFA$=""THEN100
 110 ONVAL(A$)GOSUB150,160,180,190,210,220
 120 IFA$="Q"ORA$="W"THENGOSUB240
 130 IFA$="E"ORA$="R"THENGOSUB260
 140 IFA$="T"ORA$="Y"THENGOSUB280
 142 IFA$="P"THEN2000
 145 IFA$<>"D"THEN100
 146 GOTO1000
 150 B1=B1+M:A1=SIN(B1):A2=COS(B1):GOTO170
 160 B1=B1-M:A1=SIN(B1):A2=COS(B1)
 170 A=A1:D=9:GOSUB500:A=A2:D=12:GOTO500
 180 B2=B2+M:A3=SIN(B2):A4=COS(B2):GOTO200
 190 B2=B2-M:A3=SIN(B2):A4=COS(B2)
 200 A=A3:D=15:GOSUB500:A=A4:D=18:GOTO500
 210 B3=B3+M:A5=SIN(B3):A6=COS(B3):GOTO230
 220 B3=B3-M:A5=SIN(B3):A6=COS(B3)
 230 A=A5:D=21:GOSUB500:A=A6:D=24:GOTO500
 240 PX=(PX+(A$="Q")-(A$="W"))AND63
 250 D=0:A=PX:GOTO500
 260 PY=(PY+(A$="E")-(A$="R"))AND63
 270 D=3:A=PY:GOTO500
 280 PZ=PZ+(A$="T")-(A$="Y")
 284 IFPZ<-64THENPZ=-64
 286 IFPZ>128THENPZ=128
 290 D=6:A=PZ:GOTO500
 499 REM*POKE VALUES*
 500 S=-SGN(A):A=ABS(A):IFS=-1THENS=0
 510 B=AAND255:C=(A-B)*256:POKEQ+D,S
 520 POKEQ+D+1,B:POKEQ+D+2,C:RETURN
  1000 SYSMODE,0,0,1
  1010 SYSGMOVE,0,0:SYSLINE,2,319,0
  1020 SYSLINE,2,319,199:SYSLINE,2,0,199
  1030 SYSLINE,2,0,0:SYSV:GOTO100
```







▷



```
2000 REM
2001 OPEN1,4
2010 PRINT#1
2020 PRINT#1
2030 PRINT#1,CHR$(8)
2050 DIMS%(200),T%(200),A%(200)
2060 G$=""
2070 FORN=0TO50:G$=G$+CHR$(128):NEXT
2090 PRINT#1,G$;
2100 FORN=0TO199:S%(N)=0:NEXT
2110 A=312:X=127:XD=1:TM=256:K=8192
2120 FORN=0TO199
2130 Z=NAND7:P=PEEK(K+A+Z+40*INT(N-Z))
2140 T%(N)=S%(N):A%(N)=PANDX:S%(N)=P-A%(N)
2150 A%(N)=A%(N)*XD+128+T%(N)/TM
2160 PRINT#1,CHR$(A%(N));:NEXT
2200 XD=XD*2:TM=TM/2:PRINT#1
2210 PRINT#1,G$;
2220 X=(X/2)AND255:A=A-8
2230 IFX=0THENA=A+8:X=127:XD=1:TM=256
2240 IFA>-1THEN2120
2300 FORN=0TO199
2330 A%(N)=128+S%(N)/TM
2340 PRINT#1,CHR$(A%(N));:NEXT
2350 PRINT#1
2360 CLOSE1
2370 GOTO100
4000 L=LEN(A$)
4010 FORN=1TOL:X%=XX+N-1:C$=MID$(A$,N,1):C=ASC(C$)
4020 IFC>64THENC=C-64
4030 SYSGT,X%,Y%,48,0,C:NEXT:RETURN
```

☐

Spring 1985
Personal SOFTWARE
An Argus Specialist Publication
PUTTING YOUR MICRO TO SERIOUS USE
Learn foreign languages on the Commodore 64
Create your own characters
Stretch the columns on your Spectrum
Home accounting on the BBC Micro
Great educational programs that make learning fun

No frills. No gimmicks. Just the serious business of having fun.

We know the problem only too well.

Whatever micro you have, you don't want to use it for just one thing. That would be boring.

Sometimes you want to be serious and explore its capabilities. At others you just want to cut loose and zap a few aliens or sharpen your game skills. Even try a bit of education.

Personal Software is the answer. The best of games, utilities and education.

Get to grips with it. Every quarter.

Personal SOFTWARE

A. HUKE

BASIC Disassembler

Examine the workings of ROM or mc routines with this utility which will also run on expanded Vic-20.

This machine code disassembler is written in BASIC. It contains no special POKEs or commands and will run on the Commodore 64 or any expanded VIC-20 if you make two changes for screen width.

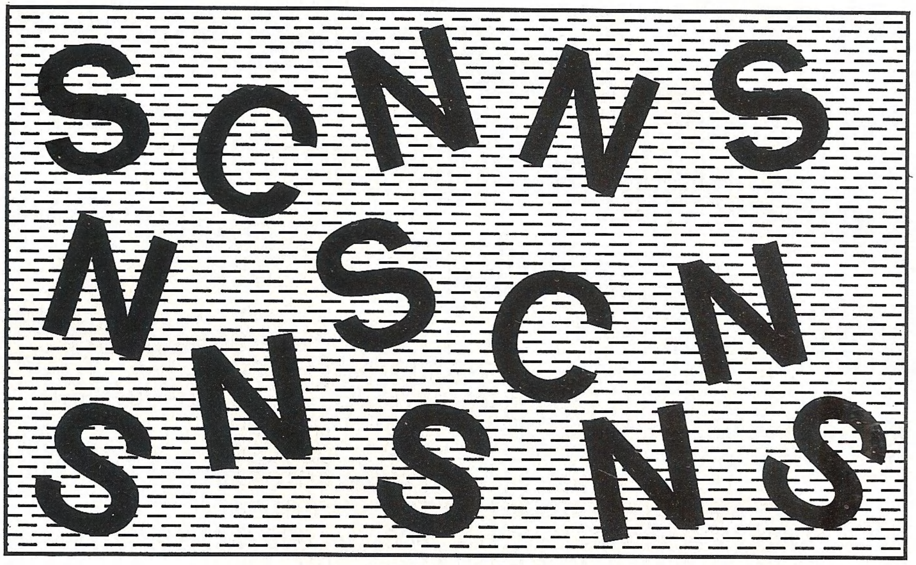
This program allows you to view in assembly language the workings of CBM 64 ROM and machine code routines included in many games and utility programs.

All numbers are displayed in hex but memory start may be entered in decimal or hex. Disassembly is started by entering S and a memory location, and halted by the space bar. Press C to continue or S to enter new start memory.

Number calculation (press N) halts disassembly. Press C or S to continue. The screen displays memory location, mnemonic operand and memory dump. The lo-byte is reversed for the operand and branches are calculated. Memory dump shows correct lo-byte order and branch offset.

VIC-20 CHANGES

The disassembler will also run on the expanded VIC-20, provided you change screen width references in lines 296 and 799.



VARIABLES USED

DIM%	*value of operand*
DN	*decimal number*
SM	*Memory being processed*
N1,N2	*used in H hex number conversion and print*
NN,H1,H2, H3,H4	*used in dec to hex conversion*
K,MM,DD	*used in hex to dec conversion*
Pn	*value of number being processed*
A$	*key input*

```
10 REM ** MACHINE CODE DISASSEMBLER **
20 DIMM%(2):GOTO860
38 REM ** CALCULATE AND PRINT HEX NUMBERS **
40 N1=INT(DN/16)+48:IFN1>57THENN1=N1+7
45 N2=DN-INT(DN/16)*16+48:IFN2>57THENN2=N2+7
50 PRINTCHR$(N1);CHR$(N2);:RETURN
58 REM ** ROUTINE FOR ONE HEX NUMBER BYTE **
60 SM=SM+1:DN=PEEK(SM):M%(1)=DN:GOSUB40:RETURN
68 REM ** ROUTINE TO REVERSE LO-BYTE HI-BYTE **
70 DN=PEEK(SM+2):M%(2)=DN:GOSUB40:DN=PEEK(SM+1):M%(1)=DN:GOSUB40:SM=SM+2:RETURN
80 REM ** DEC TO HEX CONVERSION AND PRINT **
90 H1=INT(DN/4096):H2=INT((DN-4096*H1)/256)
95 H3=INT(((DN-4096*H1)-H2*256)/16)
100 H4=(DN-4096*H1)-(H2*256)-H3*16
105 NN=H1:GOSUB120:NN=H2:GOSUB120:NN=H3:GOSUB120:NN=H4:GOSUB120:RETURN
120 NN=NN+48:IFNN>57THENNN=NN+7
130 PRINTCHR$(NN);:RETURN
150 REM ** BRANCH ROUTINE **
160 SM=SM+1:DN=PEEK(SM):M%(1)=DN:IFDN>127THEN180
170 DN=SM+DN+1:GOSUB90:RETURN
180 DN=SM-255+DN:GOSUB90:RETURN
```

```
185 REM ****
199 REM **HEX TO DEC CONVERSION **
200 K=LEN(A$):DD=4↑(K-2)
210 MM=0:FORJ=2TOK
215 N1=ASC(MID$(A$,J,1))-48:IFN1>9THENN1=N1-7
220 MM=MM+N1*(DD↑2)
230 DD=DD/4:NEXT:RETURN
290 IFSM>65535THENPRINT"MEMORY OUT OF RANGE":GOTO900
295 DN=SM:GOSUB90:PRINT"     ";:PN=PEEK(SM):M%(0)=PN:M%(1)=-1:M%(2)=-1
296 REM **PRINT 2 SPACES ↑ ON CM/64 1 SPACE FOR VIC 20**
300 IFPN=0THENPRINT"BRK";:GOTO800
310 IFPN=8THENPRINT"PHP";:GOTO800
315 IFPN=16THENPRINT"BPL ";:GOSUB150:GOTO800
320 IFPN=24THENPRINT"CLC";:GOTO800
325 IFPN=32THENPRINT"JSR ";:GOSUB70:GOTO800
330 IFPN=40THENPRINT"PLP";:GOTO800
335 IFPN=48THENPRINT"BMI ";:GOSUB150:GOTO800
340 IFPN=56THENPRINT"SEC";:GOTO800
345 IFPN=64THENPRINT"RTI";:GOTO800
350 IFPN=72THENPRINT"PHA";:GOTO800
355 IFPN=80THENPRINT"BVC ";:GOSUB150:GOTO800
360 IFPN=88THENPRINT"CLI";:GOTO800
365 IFPN=96THENPRINT"RTS";:GOTO800
370 IFPN=104THENPRINT"PLA";:GOTO800
375 IFPN=112THENPRINT"BVS ";:GOSUB150:GOTO800
380 IFPN=120THENPRINT"SEI";:GOTO800
385 IFPN=136THENPRINT"DEY";:GOTO800
390 IFPN=138THENPRINT"TXA";:GOTO800
395 IFPN=152THENPRINT"TYA";:GOTO800
400 IFPN=144THENPRINT"BCC ";:GOSUB150:GOTO800
405 IFPN=154THENPRINT"TXS";:GOTO800
410 IFPN=168THENPRINT"TAY";:GOTO800
415 IFPN=170THENPRINT"TAX";:GOTO800
420 IFPN=176THENPRINT"BCS ";:GOSUB150:GOTO800
422 IFPN=186THENPRINT"TSX";:GOTO800
425 IFPN=186THENPRINT"TSX";:GOTO800
430 IFPN=184THENPRINT"CLV";:GOTO800
435 IFPN=200THENPRINT"INY";:GOTO800
440 IFPN=202THENPRINT"DEX";:GOTO800
445 IFPN=208THENPRINT"BNE ";:GOSUB150:GOTO800
450 IFPN=216THENPRINT"CLD";:GOTO800
455 IFPN=232THENPRINT"INX";:GOTO800
460 IFPN=240THENPRINT"BEQ ";:GOSUB150:GOTO800
465 IFPN=248THENPRINT"SED";:GOTO800
470 IFPN=234THENPRINT"NOP";:GOTO800
475 IFPN=162THENPRINT"LDX #";:GOSUB60:GOTO800
480 IFPN=76THENPRINT"JMP ";:GOSUB70:GOTO800
485 IFPN=108THENPRINT"JMP (";:GOSUB70:PRINT")";:GOTO800
490 IFPN=180OR(PNAND227)=160THENPRINT"LDY ";:GOTO700
495 IFPN=44THENPRINT"BIT ";:GOSUB70:GOTO800
500 IFPN=36THENPRINT"BIT ";:GOTO700
505 IFPN=140ORPN=132ORPN=148THENPRINT"STY ";:GOTO700
510 IF(PNAND231)=6THENPRINT"ASL ";:GOTO700
515 IFPN=10THENPRINT"ASL ACC.";:GOTO800
520 IFPN=42THENPRINT"ROL ACC.";:GOTO800
530 IF(PNAND231)=38THENPRINT"ROL ";:GOTO700
535 IFPN=74THENPRINT"LSR ACC.";:GOTO800
540 IF(PNAND231)=70THENPRINT"LSR ";:GOTO700
545 IF(PNAND231)=102THENPRINT"ROR ";:GOTO700
550 IFPN=106THENPRINT"ROR ACC";:GOTO800
565 IF(PNAND243)=192THENPRINT"CPY ";:GOTO700
570 IF(PNAND243)=224THENPRINT"CPX ";:GOTO700
575 IF(PNAND7)<>5AND(PNAND7)<>6AND(PNAND7)<>1THENPRINT"???";:GOTO800
580 IFPN=137ORPN=158THENPRINT"???";:GOTO800
600 IF(PNAND227)=1THENPRINT"ORA ";
605 IF(PNAND227)=33THENPRINT"AND ";
```

```
610 IF(PNAND227)=65THENPRINT"EOR ";
615 IF(PNAND231)=70THENPRINT"LSR ";:GOTO700
620 IFPN=74THENPRINT"LSR ACC.":GOTO900
625 IF(PNAND227)=97THENPRINT"ADC ";
630 IF(PNAND231)=102THENPRINT"ROR ";
635 IFPN=106THENPRINT"ROR ACC,":GOTO900
640 IF(PNAND227)=129THENPRINT"STA ";
645 IF(PNAND231)=134OR(PN=138)THENPRINT"STX ";
650 IF(PNAND227)=161THENPRINT"LDA ";
655 IF(PNAND231)=166OR(PN=170)THENPRINT"LDX ";
660 IF(PNAND227)=193THENPRINT"CMP ";
665 IF(PNAND231)=198OR(PN=202)THENPRINT"DEC ";
670 IF(PNAND227)=225THENPRINT"SBC ";
675 IF(PNAND231)=230OR(PN=234)THENPRINT"INC ";
700 IFPN=150ORPN=182THENGOSUB60:PRINT",Y";:GOTO800:REM ** ZERO PAGE,Y **
705 IF(PNAND31)=25ORPN=190THEN:GOSUB70:PRINT",Y";:GOTO800:REM ** ABSOLUTE,Y **
710 IF(PNAND31)=1THENPRINT"(";:GOSUB60:PRINT",X)";:REM ** INDIRECT,X **
715 IF(PNAND31)=0AND(PN>159)THENPRINT"#";:GOSUB60:REM ** IMMEDIATE **
720 IF(PNAND28)=4THENGOSUB60:GOTO800:REM  ** ZERO PAGE **
725 IF(PNAND31)=9THENPRINT"#";:GOSUB60:REM ** IMMEDIATE **
730 IF(PNAND28)=12THENGOSUB70:REM ** ABSOLUTE **
735 IF(PNAND31)=17THENPRINT"(";:GOSUB60:PRINT"),Y";:REM ** INDIRECT,Y
740 IF(PNAND28)=20THENGOSUB60:PRINT",X";:REM ** ZERO PAGE,X **
755 IF(PNAND31)=29OR(PNAND31)=30THENGOSUB70:PRINT",X";:REM ** ABSOLUTE,X **
760 IFPN=188THENGOSUB70:PRINT",X";:REM **ABSOLUTE,X **
799 REM **FOR CM/64 TAB(20):FOR VIC 20 TAB(15) AND REMOVE PRINT" "; **
800 PRINTTAB(20);:FORJ=0TO2:IFM%(J)>-1THENDN=M%(J):GOSUB40:PRINT" ";
805 NEXT:PRINT
810 GETA$:IFA$<>""THENGOTO835
820 SM=SM+1:GOTO290
830 GETA$
835 IFA$="C"THEN820
840 IFA$="S"THEN900
845 IFA$="I"THEN860
846 IFA$="N"THEN950
849 GOTO830
859 REM CHARACTER "▒" IS CRSR DOWN
860 PRINT"▒▒DISASSEMBLE FROM A NEW MEMORY PRESS S"
865 PRINT"▒▒HALT DISASSEMBLEY PRESS THE SPACE BAR "
870 PRINT"▒▒CONTINUE DISASSEMBLEY PRESS C"
875 PRINT"▒▒RETURN TO INSTRUCTIONS PRESS I"
880 PRINT"▒▒ALL NUMBERS ARE LISTED IN HEX"
885 PRINT"▒▒HEX NUMBER CONVERTER PRESS N"
890 GOTO830
899 REM CHARACTER "▒" IS CRSR DOWN
900 PRINT"▒ENTER START LOCATION DEC. OR HEX"
910 INPUT"▒HEX ENTERED $NNNN 0R /NNNN     ";A$:PRINT
920 IFASC(A$)>47ANDASC(A$)<58THENSM=VAL(A$):GOTO290
925 IFLEN(A$)<>5THEN900
930 GOSUB200:SM=MM:GOTO290
950 PRINT"ENTER DEC OR HEX ($N OR /N) NUMBER "
955 INPUT"          ";A$:IFA$="S"ORA$="C"ORA$="I"THEN835
960 PRINTA$"=";
965 IFVAL(A$)>65535THEN950
970 IFASC(A$)>47ANDASC(A$)<58THENDN=VAL(A$):PRINT"$";:GOSUB90:GOTO955
975 IFLEN(A$)<2THENPRINT" FORMAT INCORRECT":GOTO950
980 GOSUB200:PRINTMM;:GOTO955
```

DISASSEMBLER

☐

THOMAS TURNBULL

Windmill Tales

A fast machine code game for the CBM64 — can Marmadule Mouse outwit the rats?

There's a magic land where mice help millers and where the rats are small but cunning. In this game you use a joystick in port 2 to control Marmaduke Mouse as he attempts to gather sacks of wheat and change them to loaves of bread for the miller.

But beware the rats who mean to make life very difficult indeed! Full instructions to play are included in the program listing.

You will find in the listing some REM statements that contain words like CRU and CRD. These refer to the following line containing reverse field characters and explain their function. Because we use an automatic system to do this, some lines are duplicated (ie 62999); when this happens just type in the FIRST line only. You may also leave out all REMs.

HOW IT RUNS

Lines	Action
0-1	*Set up variables.*
2-26	*Main routine for joystick control. Read data. Set up memory and graphics.*
57-80	*Scoring and collision detection. Game restart and win test. Random rat jumping.*
90-95	*Random sack placing.*
100-410	*Music and UDGS data.*
5000-5110	*High score table.*
9000-9015	*Bell sound.*
10000-10035	*Lose sequence and optional game end.*
20000-25000	*Graphics for windmill.*
30000-40670	*Sprite data.*
50000-50160	*Machine code data for rat movement.*
55000-55070	*Play music.*
60000	*Random multi-colour.*
61000	*Switch interrupts back to normal.*

VARIABLES USED

V	*Start of video chip*
E1	*Start of video chip + 1*
S	*Sound chip start (SID).*
JV	*Joystick status.*
FR	*Fire button.*
X,Y	*Mouse position.*
G	*For formatting score.*
CH	*Memory pointer for sprites.*

```
10 poke53281,0:poke53280,5
18 rem clr-crd
20 print"▮▮";chr$(14)
28 rem   2*crd
30 printtab(8);"▮▮Tales Of The Windmill"
38 rem   2*crd
40 printtab(12);"▮▮Instructions"
44 rem crd
45 printtab(6)"▮PLACE JOYSTICK IN PORT#2"
49 rem crd
50 print"▮This program does not need to be run"
59 rem crd crd
60 print"▮every time.The second program is        ▮independent"
69 rem crd
70 print"▮and can be run on its own":gosub63000
79 rem clr-crd
80 print"▮▮This is the story of MARMADUKE MOUSE"
90 print"Who works in a magic land as a MILLERS  helper.He has to collect the
100 print"sacks of ground wheat thrown out the"
105 print"Windmill by climbing ,jumping and
110 print"running along the windmill walls"
120 print"MARMADUKE is a magic mouse and his touch"
129 rem cru
130 print"▮changes the sacks of wheat  to loafs."
140 print"There are exceptions to this rule as
150 print"MARMADUKES magic is not that strong."
160 print"The Windmill is full of rats.
170 print"In this magic land mice are bigger than
180 print"rats.If the rats touch a sack or a loaf"
190 print"while MARMADUKE is touching it they will
199 rem cru
200 print"▮not change and he has to try again"
210 print"if he has to jump to reach a sack he
220 print"must not touch a rat on the way up or"
230 print"on the way down.At the same time no rats
239 rem cru
240 print"▮must touch a sack or loaf during a jump"
249 rem cru
250 print"▮":gosub63000
259 rem clr crd
260 print"▮"tab(8)"▮METHODS OF SCORING"
269 rem crd
270 print"▮You start of with approx 6000 points"
280 print"this is eaten away everytime a rat"
290 print"touches MARMADUKE or touches a sack,loaf
299 rem cru
300 print"▮or two rats collide"
309 rem crd
310 print"▮You gain points by touching the sacks"
320 print"and making them change into loaves."
330 print"At the start you have to change ten
340 print"loaves.This is increased by 2 loaves
350 print"every time you complete a game."
360 print"If you lose by allowing your points to
370 print"reach zero you are started of at the
380 print"same number of sacks."
390 print"A bell rings everytime a sack changes to
399 rem cru
400 print"▮a loaf."
409 rem crd
410 print"▮A tune plays on completion of mission.":gosub63000
499 rem clr-  2*crd
500 print"▮▮▮This is a fast action game as the rats"
510 print"move independently of the BASIC program."
519 rem cru
520 print"▮If you press the run/stop during a game"
```

```
530 print"the rats will still move."
540 print"Use RUN/STOP and RESTORE to switch them
550 print"off.Then type RUN <return> to restart."
559 rem crd
560 print"█The bottom of the screen contains the"
570 print"status of play in this order."
579 rem crd
580 print"█Score,Hi Score,Present Job Number"
589 rem crd
590 print"█Number Of Jobs To Do"
599 rem crd
600 print"█When the 2 right hand figures match this"
610 print"means you have only to complete the"
620 print"present job to win that stage.":gosub63000
629 rem clr   2*crd
630 print"█"tab(3)"██CONTROLLS (JOYSTICK(PORT#2))"
639 rem   2*crd
640 print"██MARMADUKE is the grey mouse in white
650 print"pants and white top hat at tbe bottom
660 print"left hand corner of the screen."
670 print"You control him with the joystick"
679 rem   2*crd
680 print"██RIGHT=move to right
689 rem crd
690 print"█LEFT=move to left
699 rem crd
700 print"█UP   =move up when under a pole"
709 rem crd
710 print"█DOWN=move down when over a pole"
719 rem crd
720 print"█FIRE BUTTON=MAKE MARMADUKE JUMP":gosub63000
729 rem clr
730 print"█Do you want to re-read instructions Y/N"
740 poke198,0:wait198,1:geta$:ifa$="y"ora$="Y"thenrun
750 ifa$="n"ora$="N"then800
760 goto740
799 rem clr-  4*crd
800 print"█████Please wait while second part loads "
809 rem   2*crd
810 clr:print"██Use SHIFT/RUN/STOP to load next part"
62999 end
62999 rem crd-rvs off
63000 printtab(6)"██press a key to continue█"
63010 poke198,0:wait198,1:poke198,0:return
ready.
```

```
0 dd=12:v=53248:e1=v+1:s=54272:poke54296,15:fori=stos+24:pokei,0:g=1000000:next
1 ch=2040:12=209:u1=211:r1=213:jo=56320:fb=16:fc=15:r%=8:13=4:u=1:d=2:goto27
2 jv=peek(jo):fr=jvandfb:jv=fc-(jvandfc)
6 ifjv=r%thenx=x+r%:pokech,r1+(peek(ch)=r1):ifx>248thenx=248
7 ifjv=13thenx=x-r%:pokech,12+(peek(ch)=12):ifx<16thenx=16
8 ifjv=0thenpokech,u1-1
9 iffr<>0ory=40then15
10 forj1=ytoy-35step-1:pokee1,j1:pokech,u1+(peek(ch)=u1):next
11 foraa=j1toy:pokee1,aa:pokech,u1+(peek(ch)=u1):next:return
15 ifx=112orx=208then17
16 return
17 ifjv=uandy=206thenfory=yto159step-1:pokee1,y:next:return
18 ifjv=uandy=158thenfory=yto112step-1:pokee1,y:next:return
19 ifjv=uandy=111thenfory=yto41step-1:pokee1,y:next:return
20 ifjv=dandy=40thenfory=yto110:pokee1,y::next:return
21 ifjv=dandy=111thenfory=yto157:pokee1,y:next:return
22 ifjv=dandy=158thenfory=yto205:pokee1,y:next
26 return
26 rem orn-bwn-lrd-gr1-gr2-lgn-lbl-gr3 clr-  4*crd blk
27 poke53281,7:cl$="████████":print"█████"tab(8)"█tales of the windmill"
28 poke53280,5:poke53276,peek(53276)or3
29 dimtt(4)
30 printchr$(142):poke52,48:poke56,48:poke56334,peek(56334)and254
```

```
31 printchr$(142):poke52,48:poke56,48:poke56334,peek(56334)and254
32 gosub55000:poke1,peek(1)and251:fori=0to511:pokei+12288,peek(53248+i):next
33 poke1,peek(1)or4:poke56334,peek(56334)or1
35 fori=12800to13015:reada:pokei,a:next
35 rem hom
36 print"█":gosub20010:ko=5:tt=1000
37 fori=13312to14207:reada:pokei,a:next
39 pokev+37,1:pokev+38,9:pokev+39,11:pokev+40,0:pokev+45,9:pokev+46,9
40 poke53272,(peek(53272)and240)+12
50 b=0:forz=49152to49283:reada:b=b+a:pokez,a:next
51 ifb<>18371thenprint"m/c data wrong check data":stop
52 poke2040,208:poke2041,217:poke2042,218:poke2043,219:poke2044,216
53 pokev+40,0:pokev+41,9:pokev+42,11:pokev+43,0:pokev+44,1:pokev+28,3
54 pokev+21,255:e%=e%+1:ife%=1thensys49152:pokev+5,111:i1%=0:j1%=0
55 pokev+23,32:pokev+29,32:pokev+11,100:pokev+10,35:pokev+16,32:gosub20000
56 sc=g+6000:poke2046,214:poke2047,214:gosub90:x=40:y=206:pokee1,y:pokev,x
57 gosub2:pokev,x:pokee1,y
58 cd=peek(v+30):if(cdand8)=8thensc=sc-150
59 if(cdand2)=2thensc=sc-75
60 if(cdand4)=4thensc=sc-125
61 if(cdand16)=16thensc=sc-250
62 ifi1%=1thenpokev+21,191:goto64
63 ifcd=129orcd=193thengosub9000:i1%=1:sc=sc+500+int(rnd(0)*ch):poke2047,215
64 ifj1%=1thenpokev+21,127:goto66
65 ifcd=65orcd=193thenj1%=1:gosub9000:sc=sc+500+int(rnd(0)*ch):poke2046,215
66 ifi1%=1andj1%=1thengosub90:x%=x%+1:i1%=0:j1%=0:pokev,40:pokee1,208:goto57
67 ifx%=kothenx%=0:gosub5000:goto55
68 sc$=right$(str$(sc),6):ifsc<=gthensc$="000000":sc=g
68 rem hom- 23*crd-blk
69 print"█████████████████████████sc=";sc$;" hi=";tt;" job="x%+1" win=";ko
70 ifsc<=gthenx%=0:i1%=0:j1%=0:gosub10000:goto55
75 ifrnd(0)<.009thenpokev+2,int(rnd(0)*r1)+fb:pokev+7,int(rnd(0)*r1)+fb
76 ifrnd(0)<.009thenpokev+9,int(rnd(0)*r1)+fb:pokev+4,int(rnd(0)*r1)+fb
80 pokev+21,255:goto57
90 pokev+12,int(rnd(0)*r1)+27:pokev+13,int(rnd(0)*100)+80
92 poke2046,r1+u:poke2047,r1+u
93 x=40:y=206:i1%=0:j1%=0:ff=peek(v+30)
94 pokev+4,int(rnd(0)*255)
95 pokev+14,int(rnd(0)*230)+20:pokev+15,int(rnd(0)*100)+80:return
100 rem 'udg's
118 rem music
120 data8,97,10,10,143,10,7,233,10,8,97,10,10,143,10,11,48,10
121 data9,104,20,10,143,20,7,223,20,9,104,20,9,104,20,10,143,20;11,48,50
122 data11,48,200,9,104,300
125 data-1,-1,-1
130 rem roof
135 data255,255,255,253,253,163,163,131
140 data255,253,189,189,157,155,139,136
150 data255,255,239,231,231,199,135,3
151 rem mice
162 data1,33,97,248,31,63,126,66
163 data4,6,7,6,79,46,62,31
164 data32,96,224,96,242,116,120,240
165 data36,102,102,126,90,102,60,24
166 data128,132,134,159,248,252,126,66
170 rem roof
180 data131,163,163,253,253,255,255,255
190 data136,139,155,157,189,189,253,255
200 data3,135,199,231,231,239,255,255
212 data4,76,76,124,142,188,142,92
213 data15,15,31,63,63,127,127,255
214 data240,240,248,252,252,254,254,255
215 data1,1,1,3,3,7,7,15
216 data128,128,128,192,192,224,224,240
220 rem brick's
230 data3,3,3,3,3,3,3,3
260 data192,192,192,192,192,192,192,192
265 data0,126,126,126,0,30,30,30
280 data0,254,254,254,0,254,254,254
290 data255,255,240,224,192,192,192,192
```

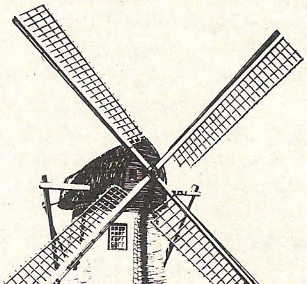
```
330 data255,255,0,0,0,0,0,0
340 data255,255,15,7,3,3,3,3
370 rem roof
380 data255,255,42,85,162,65,255,255
390 data255,255,65,162,85,42,255,255
400 data192,240,120,120,60,15,3,3
410 data3,3,15,60,120,120,240,192
5000 i1%=0:j1%=0:sc=sc-g:tt=sc:gosub55000:rem hi score
5005 z1=0:forx1=1to4:iftt>tt(x1)thenz1=x1:x1=5
5010 next:ifz1=0thenreturn
5019 rem clr-  2*crd
5020 input"███you win  enter name";n$:iflen(n$)>20thenn$=left$(n$,20)
5040 ifz1=10then5060
5050 forx1=3toz1step-1:tt(x1+1)=tt(x1):tt$(x1+1)=tt$(x1):next
5060 tt(z1)=tt:tt$(z1)=n$
5079 rem clr red
5080 print"█"tab(8)"█hi score"
5090 forx1=1to4
5094 rem crd
5095 print"█";x1;tab(10)tt(x1);tab(23);tt$(x1)
5100 next
5109 rem crd
5110 print"█press a key to restart":poke198,0:wait198,1:poke198,0:return
8999 rem bell
9000 pokes+1,130:pokes+5,9:pokes+15,30:pokes+24,15:forl8=0to1:pokes+4,21
9010 foru7=1to150:next:pokes+4,20:foru8=1to20:next
9015 foru8=stos+24:pokeu8,0:next:return
9999 rem hom-  9*crr- 10*crd
10000 print"████████████████████you lost"
10010 forbb=1to30:foraa=208to213:pokech,aa:ff=fre(0):next:next:poke53270,200
10014 rem   2*crd
10015 poke2046,r1+u:poke2047,r1+u:print"██another go (press fire button)or
10016 rem crd
10017 print"█push joystick up to stop game"
10020 jv=peek(56320)
10025 fr=jvand16:iffr=0thenreturn
10030 jv=15-(jvand15):ifjv=1thengosub61000:pokev+21,0:poke53270,200:end
10035 goto10020
19999 rem clr
20000 gosub60000:printa1$"█ABBCDEFGHIJKLMNOABCDEGHIJKLMNOPQABCDED"
20009 rem  30*crr
20010 gosub60000:printa1$"██████████████████████████████   RSRSRR"
20019 rem lbl- 30*crr pur blu
20020 print"███████████████████████████████  S█TUV█RSS"
20029 rem  30*crr pur blu
20030 print"██████████████████████████████ SS█Q P█SSR"
20039 rem  30*crr pur blk pur blu
20040 print"██████████████████████████████SRS█Q█D█P█SSS"
20049 rem  30*crr pur blu
20050 print"██████████████████████████████SRS█XXX█SRS"
20059 rem  30*crr orn blu
20060 fort=1to6:print"██████████████████████████████SRRS█S█SRSR":next
20069 rem  30*crr
20070 print"██████████████████████████████SRRSRSRSR"
20079 rem  30*crr
20080 print"██████████████████████████████RSSRSRSRR"
20089 rem  30*crr pur red
20090 print"██████████████████████████████SRS█TUV█RSS"
20099 rem  30*crr pur red
20100 print"██████████████████████████████RSS█Q P█SSR"
20109 rem  30*crr pur blk pur red
20110 print"██████████████████████████████SRS█Q█C█P█SSS"
20119 rem  30*crr pur red
20120 print"██████████████████████████████SRS█XXX█SRS"
20129 rem  30*crr
20130 print"██████████████████████████████RSSRSRSRR"
20139 rem  30*crr
20140 print"██████████████████████████████RSSRSRSRR"
20149 rem  30*crr
20150 print"██████████████████████████████RSSRSRSRR"
```

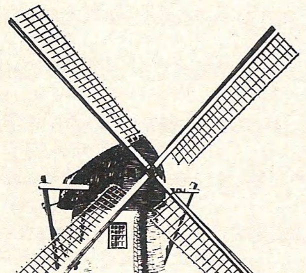
```
20159 rem  30*crr
20160 print"██████████████████████████████RSSRSRSRR"
20165 poke53270,peek(53270)or16
20170 gosub60000:printa1$"SRSRSRSRSRSRSRSRSRSRSRSRSRSRSRSRSRSRS"
20189 rem hom- 10*crd
20190 print"███████████SRSRSSRSRSRSRSSRSRSRSRSSRSRSRSSR"
20199 rem hom- 16*crd
20200 print"█████████████████STUVRSRSRSSRSRSRSSRSRSRSRSSSRSRR"
20209 rem hom  12*crr
20210 print"█":gosub60000:fort=1to22:printa1$"████████████K":next
20219 rem hom  24*crr
20220 print"█":gosub60000:fort=1to22:print"████████████████████████K":next
25000 return
```

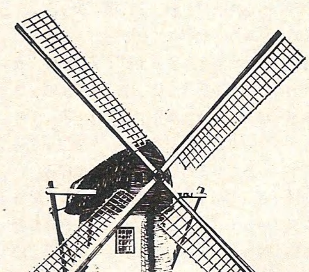
```
30000 rem mouse
30001 data0,21,0,0,21,0,0
30002 data21,0,0,21,0,0,85
30003 data64,0,63,192,0,170,192
30004 data14,170,192,2,170,0,0
30005 data170,0,0,10,0,0,170
30006 data0,2,170,0,10,42,0
30007 data24,21,128,16,21,40,0
30008 data21,2,0,17,0,0,81
30009 data0,2,73,0,2,138,0,0
30010 rem mouse
30011 data0,21,0,0,21,0,0
30012 data21,0,0,21,0,0,85
30013 data64,0,63,192,0,170,192
30014 data14,162,192,2,170,0,0
30015 data170,0,0,10,0,6,170
30016 data0,0,170,2,0,42,8
30017 data0,21,160,0,21,0,0
30018 data21,0,0,17,0,0,17
30019 data96,0,32,96,0,160,32,0
30020 rem mouse
30021 data0,85,0,0,85,0,0
30022 data85,0,0,85,0,1,85
30023 data64,0,170,0,2,130,128
30024 data0,170,0,0,130,0,0
30025 data40,0,2,170,128,10,170
30026 data160,40,170,40,96,170,9
30027 data64,85,1,0,85,0,0
30028 data85,0,0,65,0,0,64
30029 data64,2,128,16,2,128,40,0
30030 rem mouse
30031 data0,85,0,0,85,0,0
30032 data85,0,0,85,0,1,85
30033 data64,0,170,0,18,130,132
30034 data16,170,4,32,130,8,42
30035 data40,168,42,170,168,2,170
30036 data128,0,170,0,0,170,0
30037 data0,85,0,0,85,0,0
30038 data85,0,0,65,0,0,65
30039 data0,0,130,0,2,130,128,0
30040 rem mouse
30041 data1,84,0,1,84,0,1
30042 data84,0,1,84,0,5,85
30043 data0,15,252,0,14,138,0
30044 data14,170,192,14,170,0,0
30045 data168,0,0,160,0,0,170
30046 data160,0,170,128,32,168,0
30047 data8,84,0,2,84,0,0
30048 data84,0,0,84,0,0,16
30049 data0,0,40,0,0,40,0,0
30050 rem mouse
30051 data1,84,0,1,84,0,1
30052 data84,0,1,84,0,5,85
30053 data0,15,252,0,14,170,0
30054 data14,170,192,14,170,0,0
30055 data168,8,0,160,40,0,170
```

```
30056 data160,0,170,128,0,168,0
30057 data0,84,0,10,84,0,32
30058 data84,0,128,84,0,0,17
30059 data0,2,80,96,2,0,160,0
30100 rem sack
30101 data0,0,0,0,0,0,10
30102 data128,168,42,128,170,10,129
30103 data168,2,183,160,2,159,160
30104 data2,157,160,2,181,160,2
30105 data151,160,2,157,160,2,183
30106 data160,2,157,160,2,183,160
30107 data2,157,160,2,183,160,2
30108 data170,160,0,170,128,0,42
30109 data0,0,8,0,0,0,0,0
30110 rem loaf
30111 data0,0,0,0,0,0,0
30112 data0,0,0,0,0,0,0
30113 data0,0,0,0,0,0,0
30114 data0,0,0,0,0,0,0
30115 data0,0,10,170,160,42,170
30116 data168,153,153,154,102,102,102
30117 data153,153,153,102,102,102,153
30118 data153,153,102,102,102,153,153
30119 data0,0,0,0,0,0,0,0
39999 rem rat
40000 data0,192,0,1,224,0,3,254
40010 data0,3,191,0,1,252,0,0
40020 data232,0,0,204,0,1,192,0
40030 data3,236,0,7,243,128,143,224
40040 data0,135,240,0,195,248,0,97
40050 data252,0,49,252,0,31,252,0
40060 data0,204,0,0,132,0,0,130
40070 data0,1,231,0,0,0,0,32
40080 rem rat
40100 data0,0,0,0,0,0,0,0
40110 data0,0,0,0,0,0,0,0
40120 data0,0,0,0,0,0,0,0
40130 data128,0,32,128,0,112,192,240
40140 data252,97,252,94,51,254,124,31
40150 data255,240,7,255,20,7,254,28
40160 data3,254,0,1,243,224,1,0
40170 data48,3,192,0,0,0,0,32
40185 rem rat
40200 data0,0,0,0,0,0,0,0
40210 data0,0,0,0,0,0,0,0
40220 data0,0,0,0,0,0,0,0
40230 data0,0,0,24,0,1,24,0
40240 data3,120,56,6,236,254,12,61
40250 data255,184,175,255,224,225,255,128
40260 data0,127,0,0,157,0,15,0
40270 data128,8,1,0,0,14,0,32
40285 rem rat
40300 data1,128,0,1,128,0,3,192
40310 data0,30,192,0,63,224,0,47
40320 data240,0,3,240,1,135,248,7
40330 data79,248,12,63,248,24,15,252
```

```
40340 data48,23,254,96,33,255,192,0
40350 data255,128,0,127,0,0,63,0
40360 data0,30,128,0,12,64,0,4
40370 data32,0,30,240,0,0,0,32
40499 rem blade
40500 data14,0,0,31,0,28,31,128
40510 data62,31,192,126,15,224,254,7
40520 data193,252,3,131,248,1,4,240
40530 data0,136,96,0,80,0,0,32
40540 data0,8,80,0,28,136,0,63
40550 data4,0,126,14,0,252,31,0
40560 data248,63,128,240,31,192,96,15
40570 data224,0,7,224,0,3,192,0
40585 rem blade
40600 data0,63,0,0,63,0,0,63
40610 data0,0,63,0,0,63,0,0
40620 data63,0,254,63,0,254,32,0
40630 data254,32,0,254,32,0,255,255
40640 data248,0,35,248,0,35,248,0
40650 data35,248,7,227,248,7,224,8
40660 data7,224,0,7,224,0,7,224
40670 data0,7,224,0,7,224,0,0
49999 rem m/c to move rats
50000 data120,169,49,141,20,3,169,192
50010 data141,21,3,88,169,0,141,2
50020 data208,169,160,141,3,208,169,186
50030 data141,4,208,169,96,141,5,208
50040 data162,112,142,6,208,142,7,208
50050 data162,208,142,8,208,142,9,208
50060 data96,238,2,208,206,4,208,206
50070 data4,208,238,7,208,206,9,208
50080 data206,9,208,173,9,208,201,128
50090 data144,28,169,220,141,253,7,169
50100 data216,141,249,7,169,219,141,250
50110 data7,169,216,141,251,7,169,219
50120 data141,252,7,76,49,234,162,221
50130 data142,253,7,169,217,141,249,7
50140 data169,218,141,250,7,169,217,141
50150 data251,7,169,218,141,252,7,76
50160 data49,234,255,255
55000 Ko=Ko+1:restore:s=54272:rem play music
55020 poKes+24,111:poKes+23,1:poKes+4,65:poKes+5,190:poKes+3,15:poKes+2,15
55030 readh1,l1,d1:ifh1=-1thenfori=stos+24:poKei,0:next:return
55040 poKes+1,h1:poKes,l1
55050 forx1=d1-50tod1-20:poKes+6,136:next
55060 fort1=1tod1:next:poKes+1,0:poKes,0:poKes+4,0
55070 goto55020
60000 a1$=mid$(cl$,int(rnd(0)*7)+1,1):return
61000 poKe49154,49:poKe49159,234:sys49152:return
ready.
```







User Defined Graphics

How to create good-looking games, thanks to UDGs.

Most articles on graphics for the 64 concentrate on sprites to create good-looking games, whilst user defined graphics scarcely get a mention.

These useful characters, whether seen as background extras or moving objects, play an important part in any good game. Hopefully, after reading this article you'll know how to create visual effects you would have thought impossible.

CREATING UDGS

Firstly, you need a good idea of how to create UDGs. Next time you have your 64 on, look closely at the screen characters — they are not smooth but made up of 64 dots known as pixels. Each character is formed from a layout of eight by eight pixels, which are fixed graphics held in ROM and unchangeable.

However, you can change the position where the computer looks to see the patterns to display by using POKE 53272, (PEEK (53272)AND240) OR X. X is the viewing position and fixed characters are shown if X is four, five, six or seven; any other number will move the viewing position in RAM.

The most useful number is 15, as this places the viewing position farthest from your program thus giving you the most room with definitions starting at location 14336.

When you first change viewing position, all the computer sees is unused RAM and all you'll see are some oddly patterned characters. Now most games will need some form of text so the first thing to do is to copy some text characters into RAM. Lines 40-140 of Listing 1 do this by uncovering the hidden character ROM and PEEKing and POKEing in machine code.

Now you are all set to add your own characters — the sprite chapter of your manual will help here. Creating UDGs is similar in method to creating sprites in that bytes are filled in the same way. A UDG is simply eight bytes, set out as rows containing bits — Diagram 1 shows how this appears by giving an example character. Each byte total is the sum of the bit totals; to change this, simply POKE in your own numbers.

CHARACTER DEFINER

Listing 2 is a simple character definer which helps in the design and addition of bit totals. You move a flashing cursor backwards and forwards, using this space and return keys to empty or fill bytes. Pressing E will then allow you to use the byte totals at the side of your character.

Multicolour characters can also be created in the same way as with sprites. Sets of two bits are joined horizontally to make one screen pixel — this can have four colours (one character colour and three screen colours from locations 53281, 53282 and 52383). Use the following lines to switch multicolour mode on and off:

POKE 53270,PEEK(53270) OR 16.
POKE 53270,PEEK(53270) AND 239.

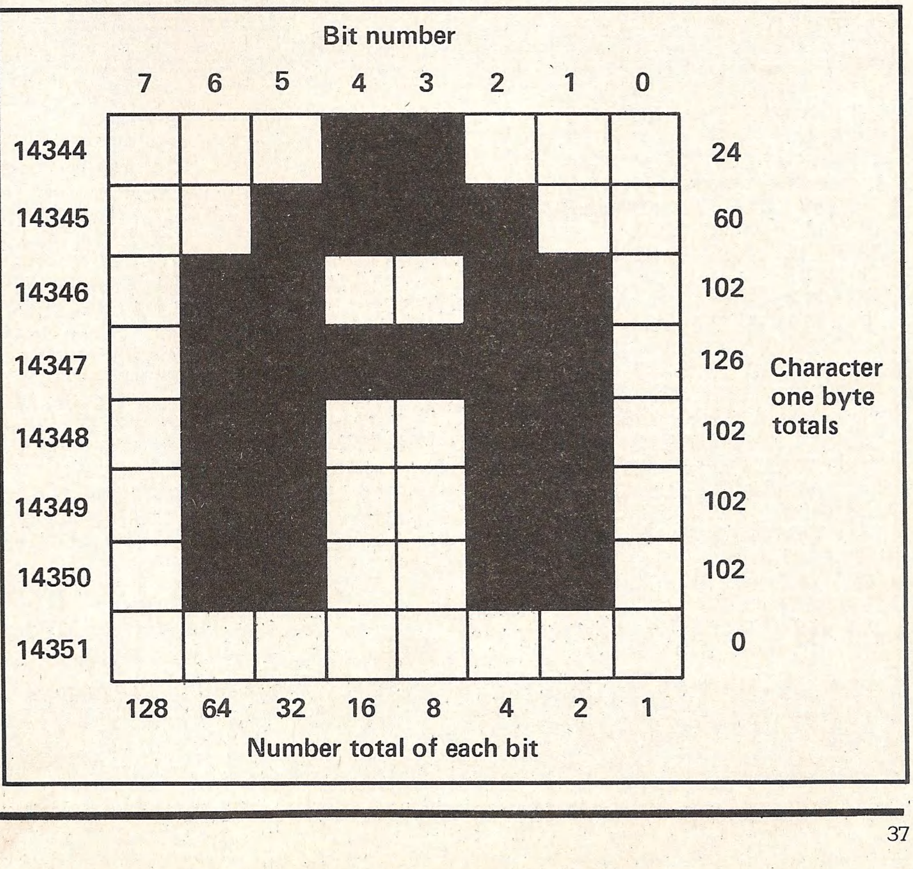
Now you have your characters, you can discover how versatile they are. UDGs can be endlessly repeated like any character so a few simple numbers and some PRINTing can be used to great effect.

Diagram 2 shows a UDG of part of a brick wall. Insignificant on its own, perhaps, but remember it can be repeated as many times as you like to form a wall of any size!

Another simple improvement is to join several characters together to make one large character. Objects can be repeated as many times as you like, instead of being limited to eight sprites.

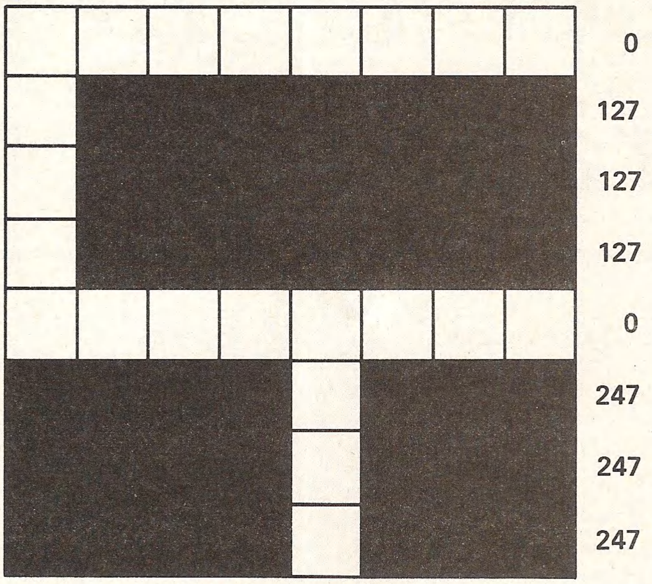
HARDWARE CHARACTERS

UDGs begin to be really impressive now. There are actually two types of UDG: hardware and software characters. The BBC and Spectrum use software characters by POKEing one byte of your design onto one byte of its hi-res screen. The 64 can do this but it is easier to use hardware characters.



In this case the computer sees your character design and displays it directly on the TV screen. Instead of having to PRINT new designs over all the old characters as with the BBC, the 64 allows you to POKE directly to the character field and effect a global change. This opens up many fantastic possibilities.

For instance, if you completely covered the screen with one repeated UDG, you could change the whole screen instantly simply by changing the defined pattern of the character. The latter part of Listing 1 shows this — displays that can be graphically copied with super-fast drawing and calculations.



GRAPHICS SCREEN

You can even use UDGs to create a small graphics screen as Listing 3 shows. This gives a 64 by 64 memory saving resolution using only 64 characters, thus allowing the rest to be used for text. Another advantage is that the screen can be easily moved as it is just a pattern of characters.

The screen is easily expandable up to a resolution of 128 by 128 or any format in steps of eight. Simply POKE out a different pattern and change the point draw routine. Many other things can be done to manipulate characters, turning them upside down, sideways or mixing up their order.

The possibilities are almost endless and all the above ideas can be used in games either to complement or replace sprites. Using similar hardware features, you could even transfer these ideas to the VIC-20. The only real limit to possible graphic display is your imagination. Use it!

```
1 REM****************
2 REM*CREATING UDGS*
3 REM*     BY       *
4 REM* DAVID REES   *
5 REM****************
9 REM* CODE FOR ROM READ *
10 FOR N=0 TO 33
20 READA:POKE49152+N,A
30 NEXT
40 DATA169,0,133,251,133,253,169,56
50 DATA133,252,169,208,133,254,160,0
60 DATA177,253,145,251,200,208,249
70 DATA230,252,230,254,165,252
80 DATA201,64,208,239,96
99 REM* RUN MACHINE CODE *
100 V=53248:POKE56334,PEEK(56334)AND254
110 POKE1,PEEK(1)AND251
120 SYS49152
130 POKE1,PEEK(1)OR4
140 POKE56334,PEEK(56334)OR1
150 POKEV+24,(PEEK(V+24)AND240)OR15
160 POKEV+32,0:POKEV+33,0
199 REM*[CLS]
200 PRINT"♥";
210 FOR N=0 TO 2
220 A$=A$+CHR$(160+N)+CHR$(157)+CHR$(17)
230 NEXT
239 REM*[UP*3]
240 A$=A$+CHR$(163)+"TTT"
250 FOR Y=0 TO 4
260 FORN=0TO39:PRINTA$;:NEXT
269 REM*[HOME]
270 PRINT"QQQ";:NEXTY
280 FORN=0TO38:PRINTA$;:NEXT
289 REM*..[HOME]
290 PRINTLEFT$(A$,10)"S"
299 REM* MAKE GRAPHICS PATTERNS *
300 X=15104:Y=31.9:Z=255:Q=170
310 FORN=0.1TO17STEP0.1
320 POKEX+M-N+0.1,85:K=N+0.1
330 FORM=NTOYSTEPN:POKEX+M,Z:POKEX+M-K,85:NEXT
340 NEXTN
360 FORN=0.1TO7STEP0.1
370 POKEX,Z:POKEX+31,0
380 FORM=NTOYSTEPN:POKEX+M,Z:POKEX+M-N,0:NEXT
390 NEXTN
400 FORN=0.1TO7STEP0.03
410 POKEX,170:POKEX+31,85
420 FORM=NTOYSTEPN:POKEX+M,Q:POKEX+M-N,85:NEXT
430 NEXTN
```

```
1 REM****************
2 REM*UDGS DESIGNER*
3 REM*     BY       *
4 REM* DAVID REES   *
5 REM****************
9 REM* CODE FOR ROM READ *
10 FOR N=0 TO 33
20 READA:POKE49152+N,A
30 NEXT
40 DATA169,0,133,251,133,253,169,56
50 DATA133,252,169,208,133,254,160,0
60 DATA177,253,145,251,200,208,249
70 DATA230,252,230,254,165,252
80 DATA201,64,208,239,96
99 REM* RUN MACHINE CODE *
100 V=53248:POKE56334,PEEK(56334)AND254
110 POKE1,PEEK(1)AND251
120 SYS49152
130 POKE1,PEEK(1)OR4
140 POKE56334,PEEK(56334)OR1
150 POKEV+24,(PEEK(V+24)AND240)OR15
160 POKEV+32,0:POKEV+33,0
199 REM*[CLS][BLUE][DWN]
200 PRINT"▒■■"
209 REM*.[RVS OFF][ORANGE]..[BLUE][RVS ON]
210 A$=" ■]||||||||■■ "
219 REM*[RVS ON]
220 PRINT"■  76543210 "
230 FOR N=0 TO 7
239 REM*[RVS ON]
240 PRINT"■"STR$(N);A$
249 REM*[UP]
250 PRINT"]",," 0"
260 NEXTN
269 REM*[RVS ON]
270 PRINT"■  76543210 "
280 PRINT"■ENTER E TO END"
285 PRINT"■     ■■"
290 S=1147:X=0:Y=0
300 P=S+X+Y*40
310 POKEP,PEEK(P)+128AND255
320 GETA$
330 POKEP,PEEK(P)+128AND255
340 IFA$=""THEN310
350 IFA$="■"THEN500
360 IFA$="■"THEN600
370 IFA$=" "THEN700
380 IFA$=CHR$(13)THEN800
390 IFA$<>"E"THEN300
399 REM*[DWN*4]
400 PRINT"■■■■"
410 END
500 X=X+1:IFX<8THEN300
510 X=XAND7:Y=Y+1:IFY<8THEN300
520 Y=YAND7:GOTO300
600 X=X-1:IFX>-1THEN300
610 X=XAND7:Y=Y-1:IFY>-1THEN300
620 Y=YAND7:GOTO300
700 B(Y)=B(Y)AND(255-2↑(7-X)):GOSUB900
710 POKEP,122
720 GOTO500
800 B(Y)=B(Y)OR2↑(7-X):GOSUB900
810 POKEP,250
820 GOTO500
899 REM*[HOME][DWN*2]
900 PRINT"■■■"
909 REM*..[LEFT]
910 FORN=0TO7:PRINT,,B(N)"■  ":NEXT
920 POKE15360+Y,B(Y)
930 RETURN
```

```
1 REM****************
2 REM*UDGS HIGH-RES*
3 REM*     BY       *
4 REM* DAVID REES   *
5 REM****************
9 REM* CODE FOR ROM READ *
10 FOR N=0 TO 33
20 READA:POKE49152+N,A
30 NEXT
40 DATA169,0,133,251,133,253,169,56
50 DATA133,252,169,208,133,254,160,0
60 DATA177,253,145,251,200,208,249
70 DATA230,252,230,254,165,252
80 DATA201,64,208,239,96
99 REM* RUN MACHINE CODE *
100 V=53248:POKE56334,PEEK(56334)AND254
110 POKE1,PEEK(1)AND251
120 SYS49152
130 POKE1,PEEK(1)OR4
140 POKE56334,PEEK(56334)OR1
150 POKEV+24,(PEEK(V+24)AND240)OR15
160 POKEV+32,0:POKEV+33,0
187 REM*DRAW CHARACTER PATTERN*
188 REM*.[RVS ON].[RVS OFF][ORANGE].
189 REM*.[BLUE][RVS ON]
190 A$=" ■ ■]         ■■ "
199 REM*[CLS][DWN].[RVS ON][BLUE]
200 PRINT"▒■ ■■          "
210 FORN=0TO7:PRINTA$:NEXT
219 REM*[RVS ON]
220 P=1106:PRINT" ■          "
230 FORX=0TO7
240 FORY=0TO7
250 POKEP+X+Y*40,192+X*8+Y
260 NEXTY,X
270 P=14336+192*8
280 FORN=0TO511:POKEP+N,0:NEXT
289 REM*[DWN]
290 PRINT"■A MINI HIGH RESOLUTION SCREEN"
299 REM*THE FUNCTION TO BE DRAWN*
300 FORC=0TOπSTEPπ/8:B=32*(π-C)/π
310 FORX=0TO63
320 Y=32-SIN(X/6+C)*B:GOSUB500
330 NEXTX,C
498 END
499 REM* DRAW POINT*
500 S=P+Y+INT(X/8)*64:A=2↑(7-(XAND7))
510 POKES,PEEK(S)ORA:RETURN
```

Code Saver

Save an area of memory to tape with this short routine.

This is a routine for the 64 that allows you to save an area of memory to tape. It will allow you to recall short areas later.

If, for instance, you have designed a sprite and want to save it here is the technique. Load the BASIC Code Saver on top of the program already in the computer. When you RUN it and type in the name under which you've saved the data or sprite, it will be recalled.

Note that when loading programs saved by this routine, you should type LOAD"",1,1

VARIABLES USED

L	*length of file name*
ADD	*starting address of SAVE*
AH%	*high byte of start of SAVE*
AL%	*low byte of start of SAVE*
A	*program length in bytes*
H%	*high byte of end of SAVE*
L%	*low byte of end of SAVE*
A$	*name of program to be saved*

```
1 REM ----------------------------
2 REM  COMMODORE-64 UTILITY
3 REM ----------------------------
4 REM
5 REM    'CODE SAVER'
6 REM
7 REM BY  JOHN MC HALE
8 REM
9 REM SLIGO,EIRE..............
10 REM
11 REM
20 PRINT"▒":PRINT"▒▒ENTER PROGRAM NAME (ALPHA CHARS ONLY) ■":PRINT:PRINT
30 INPUTA$:IFLEN(A$)>16THENA$=LEFT$(A$,16)
35 PRINT:PRINT
40 L=LEN(A$):POKE250,L
50 FORT=1TOL:P=ASC(MID$(A$,T,1))
60 IFP<65ORP>90THEN20
70 POKE20479+T,P:NEXT
100 FORT=0TOL-1:POKE1984+T,PEEK(20480+T)-64:POKE1984+T+54272,1:NEXT
102 PRINT"▒IF THE TWO PROGRAM NAMES DO NOT MATCH   THEN REDO FROM START
 "
110 PRINT:PRINT"MATCHING (Y/N)":INPUTA$:IFA$="Y"THEN121
120 GOTO20
121 PRINT:INPUT"▒STARTING ADDRESS OF SAVE";ADD:PRINT
125 AH%=(ADD/256):AL%=(ADD-(AH%*256)):POKE251,AL%:POKE252,AH%
130 PRINT"▒LENGTH OF PROGRAM (BYTES) ";:INPUTA
140 H%=(ADD+A)/256:POKE254,H%
150 L%=((ADD+A)-(H%*256)):POKE253,L%
155 PRINT:PRINT"▒END OF SAVE ADDRESS = ";H%*256+(L%-1):PRINT
160 FORT=0TO27:READA:POKE30720+T,A:NEXT
170 DATA169,1,162,1,160,1,32,186,255,165,250,162,0,160,80,32,189,255
180 DATA166,253,164,254,169,251,32,216,255,96
190 SYS(30720):PRINT:PRINT"▒PROGRAM SAVED■"
200 PRINT:PRINT"▒▒ENDING ROUTINE":END
```

DAVID WILKINSON

Galaxy Hero

Pilot the Starship Exitprise with your trusty VIC-20.

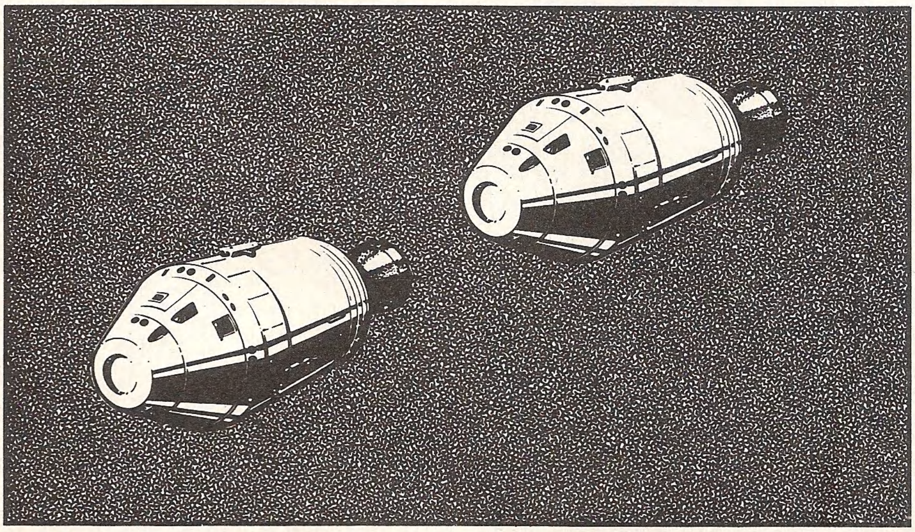
You are alone. 10,000 miles from home. Out there, somewhere, are 20 Klingon ships who'll destroy your galaxy given half a chance. You stand alone against them in this great game for the VIC-20 plus 3K of RAM.

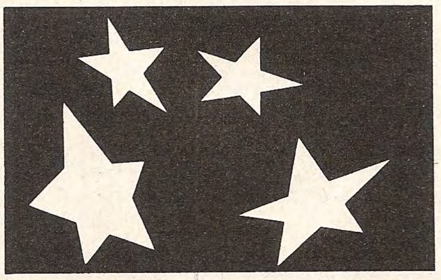
Although you don't know their positions, your ship is equipped with short and long range scans, as well as lasers and missiles. The short range scan gives you a visual display of the quadrant you're in, whilst the long range scan gives you the number of Klingons and starships in your quadrant plus the eight adjoining ones.

In each quadrant there are two numbers. The one on the left tells you the number of Klingon ships; that on the right the number of starships. The galaxy is a five by twelve grid of quadrants; each quadrant is an eight by eight grid.

You can set a time limit of up to eight minutes by which you must shoot a ship (uses the VIC clock) — this also affects your energy level. There are also eight difficulty levels, with one the hardest.

Once you have shot a ship, enter it onto the log record by a long range scan of the quadrant. Now, boldly go into space!





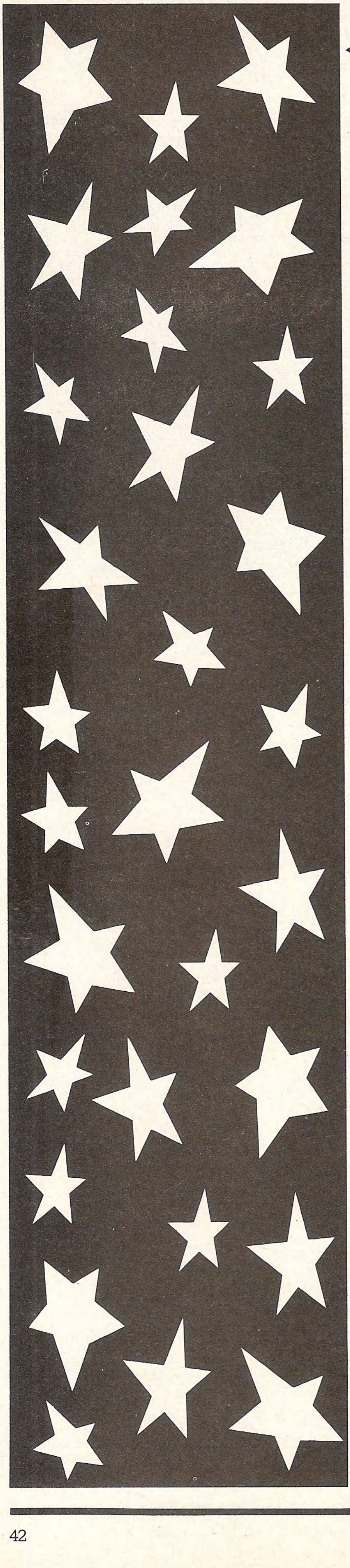
HOW IT RUNS

1-16	*sets up variables*
17-55	*menu*
100-135	*move*
150-202	*short range scan*
250-298	*long range scan*
300-350	*laser*
400-445	*missile*
450-470	*log record*
500-530	*current status*
700-810	*messages*
1000-1230	*sound effects*
1500-1600	*messages*
9100-9200	*title pages*

VARIABLES USED

L$	*quadrant display*
KL	*quadrant number of Klingons*
SH	*quadrant number of starship*
K1,K2,S1,S2	*coordinates*
FU	*energy*
MI	*missiles*
KD	*Klingons*
SD	*starships*
X,Y	*spaceship's coordinates*
Q	*quadrant*

```
0 GOTO9100
1 DIML$(60),KL(20),SH(4),K1(4),K2(4),S1(4),S2(4):FU=2000:MI=4
2 FORI=1TO60:L$(I)="**":NEXT:KD=20:SD=4:TI$="000000"
3 FORI=0TO20:KL(I)=INT(RND(1)*60)+1:NEXT
4 FORI=0TO4:SH(I)=INT(RND(1)*60)+1:NEXT
6 X=INT(RND(1)*40)+1:Y=INT(RND(1)*96)+1
7 POKE36878,15
8 SO=36874
15 Q=INT((X-1)/8)+1+INT((Y-1)/8)*5
16 FORI=1TO20:IFKL(I)=0THENNEXT:GOTO700
17 PRINT"▀█":POKE198,0
18 PRINT
19 PRINT
20 PRINT"   0=MOVE"
22 PRINT"   1=SHORT RANGE"
24 PRINT"   2=LONG RANGE"
26 PRINT"   3=LASER"
28 PRINT"   4=MISSILE"
30 PRINT"   5=SHIP'S LOG"
32 PRINT"   6=CURRENT STATUS"
34 PRINT"███X=";X,"Y=";Y:PRINT"QUADRANT=";Q
35 FORI=1TO20:IFKL(I)=QTHEN37
36 NEXTI:GOTO40
37 IFKW=1THEN40
38 PRINT"██DANGER KLINGONS!!":PRINT"YOU HAVE LOST";
39 I=INT(RND(1)*100)+50:FU=FU-I:PRINTI:PRINT"UNITS OF FUEL█":KW=1
40 GETA$:IFA$=""THEN40
42 IFTI>TM*3600THENGOTO800
45 IFA$="0"THEN100
```



```
46 IFA$="1"THEN150
47 IFA$="2"THEN250
48 IFA$="3"THEN300
49 IFA$="4"THEN400
50 IFA$="5"THEN450
51 IFA$="6"THEN500
55 GOTO40
100 PRINT"█"
101 PRINT"INPUT COORDINATES OF  REQUIRED POSITION"
105 INPUT"X COORDINATE=";X1
110 INPUT"Y COORDINATE=";Y1
115 IFX1>40ORX1<1ORY1>96ORY1<1THENPRINT"██YOU HAVE FALLEN OUT OFTHE UNIVERSE":EN
D
120 FU=FU-INT((((X1-X)↑2+(Y1-Y)↑2)↑.5)/2)
125 X=X1:Y=Y1
130 IFFU<0THENPRINT"██YOU HAVE RUN OUT OF   FUEL":END
135 GOTO15
150 PRINT"█████";
155 PRINT"     12345678"
156 PRINT"    ┌┴┴┴┴┴┴┴┴"
157 PRINT"   1┤        "
158 PRINT"   2┤        "
159 PRINT"   3┤        "
160 PRINT"   4┤        "
161 PRINT"   5┤        "
162 PRINT"   6┤        "
163 PRINT"   7┤        "
164 PRINT"   8┤        "
165 X1=X-INT((X-1)/8)*8:Y1=Y-INT((Y-1)/8)*8
166 POKE7771+X1+(Y1*22),65
168 KL=0:FORI=1TO20:IFKL(I)=QTHENKL=KL+1
170 NEXT:IFKL=0THEN185
172 FORI=1TOKL
174 K1(I)=INT(RND(1)*8)+1:K2(I)=INT(RND(1)*8)+1:IFK1(I)=X1ORK2(I)=Y1THEN174
176 FORL=0TOI-1:IFK1(I)=K1(L)ORK2(I)=K2(L)THEN174
178 NEXTL,I
180 FORI=1TOKL:POKE7771+K1(I)+(K2(I)*22),88:NEXT
185 SH=0:FORI=1TO4:IFSH(I)=QTHENSH=SH+1
186 NEXT:IFSH=0THEN202
188 FORI=1TOSH
190 S1(I)=INT(RND(1)*8)+1:S2(I)=INT(RND(1)*8)+1:IFS1(I)=X1ORS2(I)=Y1THEN190
192 FORL=0TOI-1:IFS1(I)=S1(L)ORS2(I)=S2(L)THEN190
194 NEXTL:IFKL=0THEN200
196 FORL=1TOKL:IFS1(I)=K1(L)ORS2(I)=K2(L)THEN190
198 NEXTL
200 NEXTI:FORI=1TOSH:POKE7771+S1(I)+(S2(I)*22),81:NEXT
202 POKE198,0:WAIT198,1:POKE198,0:GOTO15
250 PRINT"█"
252 PRINT"█████      ┌──┬──┬──┐"
253 PRINT"      │  │  │  │"
254 PRINT"      ├──┼──┼──┤"
255 PRINT"      │  │  │  │"
256 PRINT"      ├──┼──┼──┤"
257 PRINT"      │  │  │  │"
258 PRINT"      └──┴──┴──┘"
260 Q1=Q:GOSUB294:PRINT"███████████████████";L$(Q1)
262 IFX<9ORY<9THEN266
264 Q1=Q-6:GOSUB294:PRINT"██████████████";L$(Q1)
266 IFY<9THEN270
268 Q1=Q-5:GOSUB294:PRINT"█████████████████";L$(Q1)
270 IFY<9ORX>32THEN274
272 Q1=Q-4:GOSUB294:PRINT"████████████████████";L$(Q1)
274 IFX<9THEN278
276 Q1=Q-1:GOSUB294:PRINT"████████████████";L$(Q1)
278 IFX>32THEN282
280 Q1=Q+1:GOSUB294:PRINT"██████████████████████";L$(Q1)
282 IFX<9ORY>88THEN286
284 Q1=Q+4:GOSUB294:PRINT"██████████████████";L$(Q1)
286 IFY>88THEN290
288 Q1=Q+5:GOSUB294:PRINT"████████████████████";L$(Q1)
290 IFX>32ORY>88THEN293
292 Q1=Q+6:GOSUB294:PRINT"████████████████████████";L$(Q1)
293 POKE198,0:WAIT198,1:POKE198,0:GOTO15
294 KL=0:FORI=1TO20:IFKL(I)=Q1THENKL=KL+1
295 NEXT
296 SH=0:FORI=1TO4:IFSH(I)=Q1THENSH=SH+1
297 NEXT
298 L$(Q1)=RIGHT$(STR$(KL),1)+RIGHT$(STR$(SH),1):RETURN
300 PRINT"██INPUT COORDINATES OF  LASER BEAM":KW=0
305 INPUT"X-COORDINATE";X2
310 INPUT"Y-COORDINATE";Y2
312 IFX2<0ORX2>8ORY2<0ORY2>8THEN316
```

```
314 GOTO320
316 PRINT"▒▒ IRREGULAR COMMAND.    PLEASE TRY AGAIN"
318 FORI=1TO3000:NEXT:GOTO300
320 PRINT"▒INPUT ENERGY REQUIRED FOR BEAM"
322 INPUTL:IFFU-L<0THENPRINT"▒YOU HAVEN'T ENOUGH    ENRGY":GOTO322
323 IFL<0THENPRINT"▒DON'T BE DAFT!!!":GOTO322
324 FU=FU-L:IFX2=X1ANDY2=Y1THENGOSUB1200:PRINT"▒YOU HAVE DESTROYED    YOURSELF!!
!":END
325 IFKL=0THEN330
326 FORI=1TOKL:IFK1(I)=X2ANDK2(I)=Y2THEN335
328 NEXTI
330 IFSH=0THEN350
332 FORI=1TOSH:IFS1(I)=X2ANDS2(I)=Y2THEN340
334 NEXTI:GOTO350
335 P=INT(((X1-K1(I))↑2+(Y1-K2(I))↑2)↑.5)*20
336 IFL<PTHENGOSUB1000:GOTO15
337 FORI=1TO20:IFKL(I)=QTHENKL(I)=0:GOTO339
338 NEXTI
339 GOSUB1200:KD=KD-1:GOTO15
340 P=INT(((X1-S1(I))↑2+(Y1-S2(I))↑2)↑.5)*20
341 IFL<PTHENGOSUB1000:GOTO15
342 FORI=1TO4:IFSH(I)=QTHENSH(I)=0:GOTO346
343 NEXTI
346 FU=FU+500:TI$="000000"
347 GOSUB1200:SD=SD-1:GOTO15
350 GOSUB1000:GOTO15
400 PRINT"♥":KW=0
410 IFMI=0THENPRINT"▒▒YOU HAVE NO MISSILES    LEFT!!!":FORI=1TO3000:NEXT:GOTO15
412 IFRND(1)<.15THENPRINT"▒MALFUNCTION-MISSILE   SELF DESTRUCTED!!!":GOTO1500
415 MI=MI-1
420 FORI=1TO20:IFKL(I)=QTHENKL(I)=0:KD=KD-1
425 NEXTI
430 FORI=1TO4:IFSH(I)=QTHENSH(I)=0:FU=FU+500:SD=SD-1:TI$="000000"
435 NEXTI
440 PRINT"▒EVERYTHING IN SIGHT   ELIMINATED!!!"
442 GOSUB1210:GOTO15
445 FORI=1TO3000:NEXT:GOTO15
450 PRINT"♥";
455 FORI=1TO60:PRINT"  "L$(I);
460 IFI/5=INT(I/5)ANDI<>60THENPRINT"                         ";
470 NEXTI:PRINT"▒":POKE198,0:WAIT198,1:POKE198,0:GOTO15
500 PRINT"♥▒▒    CURRENT STATUS"
505 PRINT"    ────────────────"
510 PRINT"▒▒ FUEL="FU
512 PRINT" MISSILES="MI
515 PRINT" KLINGONS="KD
520 PRINT" STAR SHIPS="SD
525 PRINT" TIME EXPIRED="LEFT$(RIGHT$(TI$,4),2)":"RIGHT$(TI$,2)
530 POKE198,0:WAIT198,1:POKE198,0:GOTO15
700 PRINT"♥▒▒    WELL DONE"
705 PRINT"▒▒YOU HAVE KILLED ALL   THE KLINGONS."
710 PRINT"▒▒▒▒":END
800 PRINT"♥▒▒YOU HAVE BEEN IN SPACETOO LONG INBETWEEN    RESTS▒▒▒▒▒"
810 END
1000 POKESO,220:FORI=1TO700:NEXT
1010 FORI=15TO0STEP-1:POKE36878,I:FORL=1TO100:NEXTL,I:POKESO,0:POKE36878,15:RETU
RN
1200 POKESO,220:FORI=1TO700:NEXT:POKESO,0
1210 POKESO+3,170:FORI=1TO2000:NEXT:
1220 FORI=15TO0STEP-1:POKESO+4,I:FORL=1TO100:NEXTL,I:POKESO+3,0:POKESO+4,15
1230 RETURN
1500 GOSUB1210
1505 MI=MI-1:IFRND(1)<.5THEN1550
1510 PRINT"▒YOUR SHIP IS UNDAMAGEDAND YOU CAN PROCEED   WITH YOUR MISSION!!!"
1520 FORI=1TO2000:NEXT:GOTO15
1550 IFRND(1)<.1THEN1600
1560 PRINT"▒YOUR SHIP WAS DAMAGED BY THE BLAST AND YOU  HAVE LOST ENERGY!!!"
1570 FU=FU-INT(RND(1)*400)+200:FORI=1TO3000:NEXT:GOTO15
1600 PRINT"▒YOUR SHIP HAS BEEN    DESTROYED BY THE BLAST":END
9100 POKE36879,25
9101 PRINT"▒▒WELCOME ABOARD THE    STARSHIP EXITPRISE!!"
9102 PRINT"▒YOU ARE STRANDED 10000LIGHT-YEARS FROM EARTH▒▒YOUR MISSION■:"
9103 PRINT"DESTROY THE KLINGONS  IN YOUR GALAXY"
9104 PRINT"▒▒     HIT A KEY":POKE198,0:WAIT198,1:POKE198,0
9110 PRINT"♥▒▒TIME LIMIT IN MINS 1-8"
9115 PRINT" 1=HARD 8=EASY"
9120 GETA$:IFA$=""THEN9120
9130 A=VAL(A$):TM=A
9140 IFTM<1ORTM>8THEN9120
9145 PRINT"▒KLINGON----♠":PRINT"STARSHIP---●":PRINT"YOUR SHIP--♠"
9150 PRINT"▒▒▒    HIT A KEY":POKE198,0:WAIT198,1:POKE198,0
9200 GOTO1
```

JOE SAYONES

The Bells! The Bells!

Quasimodo, the Hunchback of Notre Dame, may be ugly but with your help he can conquer all.

Quasimodo, the disfigured and deformed bell-ringer of Notre Dame, has been the subject of various programs for different computers. Type in this graphic game and Vic-20 owners can join in the fun.

He never had much going for him: physical wreck, thankless job pulling those bells and, to cap it all, the supreme misfortune of falling in love with the beautiful Esmeralda. Hopeless case.

THE GAME

Thing's don't seem to have got much better in this game either. He has to pull the cathedral bells but also avoid the missiles thrown by the unkind peasants below the walls.

To help him out, use CTRL to jump over the missiles and / to move right. Good luck — at least you don't look like him as well, do you?

HOW IT RUNS

10-60	*set the variables*
100-300	*move Quasimodo*
300-400	*move the barrels*
400-500	*move the sentries*
500-600	*move the arrows*
700-800	*move to next phase*
1000-2000	*phase choice routine*
4000-5000	*dying routine*
5000-6000	*how well you did*

VARIABLES USED

SC	*score*
H	*high*
S	*speaker 3*
Q	*beginning of screen memory*
W	*beginning of colour memory*
P	*position of man*
PI	*direction of movement*
A	*arrow*
B	*barrel*
Y	*character you are*
Z	*phase (1-14)*

```
10 PRINT"■PLEASE WAIT.."
20 FORT=7168TO7168+8*11-1:READS:POKET,S:NEXT
25 FORT=7424TO7432:POKET,0:NEXT
30 PRINT"■■■NOW LOAD PART TWO"
100 DATA0,0,0,65,254,65,0,0
110 DATA0,0,60,126,126,126,126,60
120 DATA64,64,64,64,92,92,72,92,126,92,92,28,20,20,20,54
130 DATA0,0,0,0,92,92,72,92,126,92,92,92,84,84,20,54
140 DATA24,36,34,36,24,60,124,124
150 DATA124,60,60,36,38,34,34,51
160 DATA124,60,60,36,36,36,36,54
170 DATA0,0,0,0,0,0,0,0
180 DATA253,253,253,0,239,239,239,0
```

```
5 DIMS1$(3),S2$(3)
10 POKE36869,255:POKE36879,59:POKE36878,15:PRINTCHR$(8):CO=30720
20 Q=7680:W=38400:S=36876:FORT=1TO3:READS1$(T),S2$(T):NEXT
30 SC=0:L=3:Z=1
40 P=7790:P1=0:Y=7:F=1
50 GOSUB1000:POKEP,Y:POKEP+CO,0:POKEP-22,6:POKEP-22+CO,0
100 PRINT"■■SCORE";SC
110 IFPEEK(P-22)<>6THEN4000
120 IFPEEK(P)<>YTHEN4000
```

```
130 IFPEEK(P+22)=2THEN4000
140 IFP>7790+17THEN700
150 IFP1<0THENP1=P1+44:GOTO180
160 POKES,0:P1=0:IFPEEK(197)=30THENP1=P1+1:POKES,135
170 IFPEEK(653)>3THENP1=P1-22:POKES,250
180 N=P+P1
190 IFPEEK(N)<6THEN4000
200 IFPEEK(N-22)=0THEN4000
210 IFPEEK(N+22)=2THEN4000
220 Y=Y+1:IFY=9THENY=7
230 POKEP,32:POKEP-22,32
240 P=P+P1:POKEP+CO,0:POKEP,Y:POKEP-22+CO,0:POKEP-22,6
300 IFZ=2ORZ=3ORZ=5ORZ=6ORZ=8ORZ=9ORZ=11ORZ=14THEN400
310 IFB<>22THEN340
320 IFINT(1+6*RND(0))<3THENB=B-2
330 IFB=22THEN400
340 B=B-1:POKEQ+B+1+22*5,32:IFB=0THENB=22:GOTO400
350 POKEW+B+22*5,2:POKEQ+B+22*5,1
400 IFZ=1ORZ=2ORZ=3ORZ=4ORZ=7ORZ=8ORZ=9ORZ=10ORZ=12THEN500
410 X=X+1:IFX<>2THENPRINTS2$(F):GOTO500
420 PRINTS1$(F)
430 POKES,INT(127+50*RND(0))
440 F=INT(1+3*RND(0))
450 X=0
500 IFZ=1ORZ=2ORZ=4ORZ=5ORZ=8ORZ=10ORZ=14THEN600
510 IFA<>22THEN540
520 IFINT(1+6*RND(0))<2THENA=A-2
530 IFA=22THEN600
540 A=A-1:POKEQ+A+1+22*3,32:IFA=0THENA=22:GOTO600
550 POKEW+A+22*3,4:POKEQ+A+22*3,0
600 GOTO100
700 POKES+1,135:FORT=50TO59:POKE36879,T:FORY=1TO25:NEXT:NEXT:POKES+1,0
710 Z=Z+1:SC=SC+50*Z:GOTO40
1000 IFZ=15THENFORT=1TO1000STEP25:PRINT"▮▮SCORE";SC+T:NEXT:SC=SC+T:Z=1
1005 A=22:B=22:X=0
1010 IFZ=1ORZ=7THEN1100
1015 IFZ=14THEN1600
1020 IFZ=2ORZ=4ORZ=3ORZ=5ORZ=6THEN1200
1030 GOTO1300
1100 PRINT"▮▮▮▮▮▮▮▮▮JJJJJJJJJJJJJJJJJJJJJJ":GOTO1400
1200 PRINT"▮▮▮▮▮▮▮▮▮JJJ▮B▮JJJ▮B▮JJJ▮B▮JJJJJJ":GOTO1400
1300 PRINT"▮▮▮▮▮▮▮▮▮JJ▮B▮JJ▮B▮J▮B▮J▮JJ▮B▮J▮B▮J▮B▮JJJJJJ"
1400 FORT=1TO8
1410 PRINT"▮▮JJJJJJJJJJJJJJJJJJJJJJ":NEXT
1420 IFZ=5ORZ=6ORZ>12THENFORT=1TO3:PRINTS2$(T):NEXT
1500 A=22:B=A:X=0:RETURN
1600 PRINT"▮";TAB(18);"▮JJJ▮▮▮▮J▮▮▮▮JJ▮▮▮▮JJ▮▮▮▮JJ▮▮▮▮JJ"
1610 GOTO1400
4000 POKEP,32:POKEP-22,32
4010 POKEN,Y:POKEN-22,6:POKEN+CO,0:POKEN-22+CO,0
4020 POKES,128:FORT=1TO1000:NEXT:POKES,0
4030 L=L-1:IFL>0THENGOTO40
5000 PRINT"▮▮SCORE IS";SC
5010 IFSC>HTHENPRINT"▮▮▮SETTING A NEW HIGH":H=SC
5020 PRINT"▮▮▮▮HIGH IS ";H
5030 PRINT"▮▮▮▮▮▮▮▮PRESS A KEY TO GO ON"
5040 POKE198,0:WAIT198,1:GOTO30
6000 DATA"▮▮▮▮▮▮▮▮▮▮▮▮D▮▮E▮J","▮▮▮▮▮▮▮▮▮▮▮▮B▮▮C▮J"
6010 DATA"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮D▮▮E▮J","▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮B▮▮C▮J"
6020 DATA"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮D▮▮E▮J","▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮B▮▮C▮J"
```

ALAN BLACKHAM

MUSIC AID

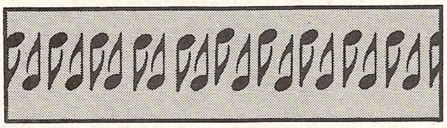
Been looking for an easy way to play music on the VIC-20? You need look no further.

In the bad old days, writing music on the VIC-20 was slow and boring. But now, thanks to Music Aid, those days have gone forever.

The program starts by displaying a musical stave with letters indicating the positions of the notes. You find the note you want and then enter the appropriate letter.

When you've entered all the notes you want, just press the back arrow on the top left of the keyboard and go on to enter the spacing and timing by pressing the space bar. If you go wrong, just press the back arrow and you can start again.

After all that: listen to your creation and list the tune data if you're happy with it. Go to it, maestro!



HOW IT RUNS

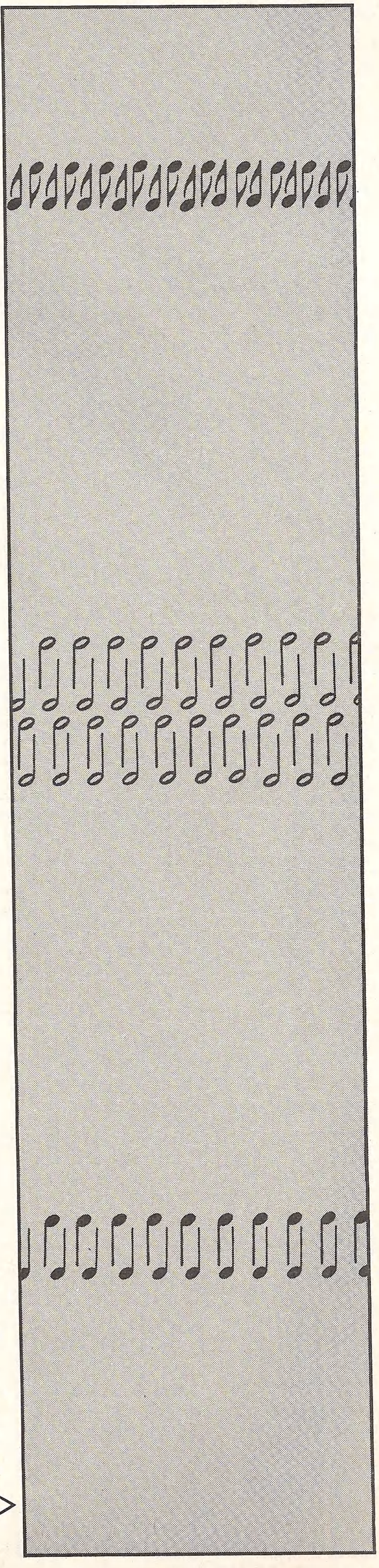
20-80	*print instructions*
100-195	*print available notes*
200-295	*enter notes*
300-460	*input timing*
500-595	*play tune*
600-660	*input option*
700-930	*list data*
1000-9000	*end*
9000-9070	*wait for key to be pressed*
9100-9160	*invalid input*
40000-40040	*data for notes*

VARIABLES USED

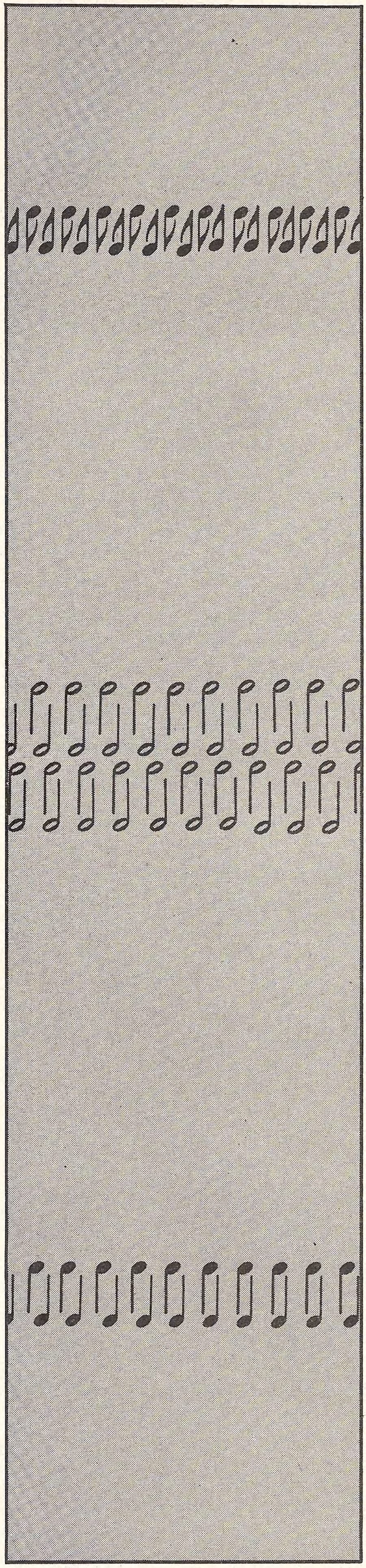
N(1-100)	*notes*
D(1-100)	*length of notes*
I(1-100)	*time between notes*
X	*current note*
N$	*input*
R,I,J	*FOR loops*
A$	*inputs*

```
1 REM     *******************
2 REM     * ALAN BLACKHAM'S *
3 REM     *     MUSIC AID   *
4 REM     *    (05/09/83)   *
5 REM     *******************
7 REM
10 POKE 36879,29:POKE 808,127
15 DIM N(100),D(100),I(100)
17 X=0
20 REM
22 REM ****** INSTRUCTIONS ******
25 REM
30 PRINT"■■■   MUSIC AID(C)1983"
35 PRINT"■■   BY ALAN BLACKHAM"
40 PRINT"■■■                    ■";
45 PRINT"■                    ■"
50 PRINT"■■FIRST ENTER THE NOTES"
55 PRINT"■FOR YOUR TUNE."
60 PRINT"■■ENTER THEM BY THE"
65 PRINT"■LETTER UNDER THE NOTE"
70 PRINT"■YOU NEED."
75 PRINT"■■"
80 GOSUB 9000
100 REM
103 REM ****** PRINT LINES ******
105 REM
107 PRINT"■■■ ───── ───── ───── ■";
110 PRINT"■ THE NOTES AVALIABLE:"
115 PRINT"■■ ───── ───── ───── ■";
117 PRINT"                    ■O■■"
120 PRINT"                   ■N■■"
125 PRINT"                  ■M■■"
130 PRINT"─────────────────■L■■────────";
135 PRINT"                ■K■■"
140 PRINT"───────────────■J■■──────────";
145 PRINT"              ■I■■"
150 PRINT"─────────────■H■■────────────";
155 PRINT"            ■G■■"
160 PRINT"───────────■F■■──────────────";
165 PRINT"          ■E■■"
170 PRINT"─────────■D■■────────────────";
174 PRINT"        ■C"
176 PRINT"■■     ─■B■■─"
178 PRINT"      ■A"
```

```
185 PRINT"■■ ▁▁▁▁▁ ▁▁▁▁▁ ▁▁▁▁▁ ■";
190 PRINT"■  PLEASE ENTER NOTES"
195 PRINT"■  (ENTER '←' TO END)"
200 REM
203 REM ****** ENTER NOTES ******
205 REM
210 POKE 198,0:X=X+1:N$=""
220 PRINT"■■■■■■■■■■■■■■■■■■■■■■"
230 PRINT"■NOTE "X;
232 INPUT"■";N$
235 IF LEN(N$)<>1 THEN GOSUB 9100:X=X-1:GOTO 100
236 IF N$="←" THEN 300
237 N=ASC(N$)
238 IF N<65 OR N>79 THEN GOSUB 9100:X=X-1:GOTO 100
250 RESTORE
255 READ A$,A:IF A$<>N$ THEN 255
256 N(X)=A
260 PRINT"■■ ▔▔▔▔▔ ▔▔▔▔▔ ▔▔▔▔▔ ■";
265 POKE 36876,220
266 FOR R=1 TO 50:NEXT R
267 POKE 36876,0
270 IF X<100 THEN 100
275 PRINT"■■■■■■■   OUT OF MEMORY !■■"
280 FOR I=128 TO 245
285 POKE 36877,I:NEXT 1
267 POKE 36876,0
270 IF X<100 THEN 100
275 PRINT"■■■■■■■   OUT OF MEMORY !■■"
280 FOR I=128 TO 245
285 POKE 36877,I:NEXT I
290 POKE 36877,0
295 GOSUB 9000
300 REM
303 REM ****** INPUT DURATION ******
305 REM
310 PRINT"■■■ ▁▁▁▁▁ ▁▁▁▁▁ ▁▁▁▁▁ ■";
320 PRINT"■  GET TIMING CORRECT"
330 PRINT"■■ ▔▔▔▔▔ ▔▔▔▔▔ ▔▔▔▔▔ ■"
335 PRINT"■■PRESS THE SPACE BAR"
340 PRINT"■FOR EACH NOTE TO SET"
345 PRINT"■THE DURATION ."
350 PRINT"■■PRESS '←' TO RE-START"
355 PRINT"■■■ ▁▁▁▁▁ ▁▁▁▁▁ ▁▁▁▁▁ ■"
360 PRINT"■■■THERE ARE■";X-1;"■NOTES"
400 FOR I=1 TO X-1
410 POKE 198,0:TI$="000000"
420 GET A$:IF A$=""THEN 420
425 I(I)=TI
430 IF A$="←" THEN 300
435 TI$="000000"
440 POKE 36876,N(I)
450 IF PEEK(197)<>64 THEN 440
460 D(I)=TI:POKE 36876,0:NEXT I
500 REM
503 REM ****** HEAR TUNE ******
505 REM
507 PRINT"■";
510 FOR J=1 TO 154
515 PRINT"■■ ■";:NEXT J
518 PRINT"■■ ▁▁▁▁▁ ▁▁▁▁▁ ▁▁▁▁▁ ■";
```

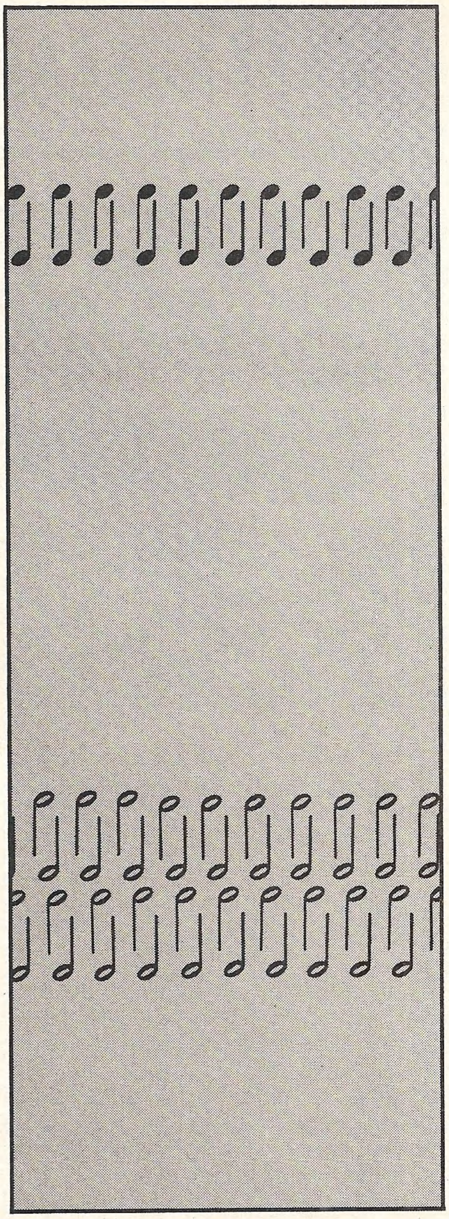


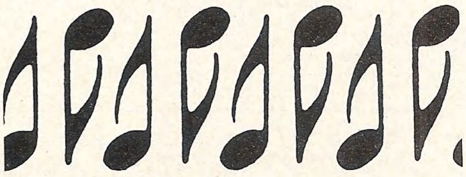
▷



```
520 PRINT"■   PRESS ANY KEY TO"
525 PRINT"■   HEAR THE TUNE !"
530 PRINT"▓▓ ━━━━━━ ━━━━━━ ━━━━━━ ■"
535 POKE 198,0
540 GET A$:IF A$=""THEN 540
550 FOR I=1 TO X-1
560 POKE 36876,N(I)
570 TI$="000000"
580 IF TI<D(I) THEN 580
590 POKE 36876,0
591 TI$="000000"
593 IF TI<I(I) THEN 593
595 NEXT I
600 REM
603 REM ****** ENTER OPTION ******
605 REM
607 PRINT"▓▓▓▓▓▓----------------------"
610 PRINT"■DO YOU WANT TO :"
620 PRINT"▓▓ 1..LIST DATA"
622 PRINT"▓ 2..RE-ENTER NOTES"
625 PRINT"▓ 3..RE-ENTER TIMING"
630 PRINT"▓▓▓PLEASE ENTER 1,2 OR 3."
635 PRINT"▓---------------------"
640 POKE 198,0
650 GET A$:IF A$<>"1"AND A$<>"2"AND A$<>"3" THEN 650
660 ON VAL(A$)' GOTO 700,17,300
700 REM
703 REM ****** LIST DATA ******
705 REM
710 PRINT"▓■THE PROGRAM TO USE"
712 PRINT"THE DATA SHOULD BE"
715 PRINT"LIKE THIS :▓▓"
720 PRINT" 10 READ N,D,I"
725 PRINT" 20 IF N=-1 THEN END"
730 PRINT" 30 POKE 36876,N"
735 PRINT" 40 TI$='000000'"
740 PRINT" 50 IF TI<D THEN 50"
745 PRINT" 60 POKE 36876,0"
750 PRINT" 70 TI$='000000'"
755 PRINT" 80 IF TI<I THEN 80"
760 PRINT" 90 GOTO 10"
770 PRINT"▓▓■PRESS ANY KEY TO SEE  THE DATA  "
775 POKE 198,0
780 GET A$:IF A$="" THEN 780
800 NL=0:N=0:PRINT"▓"
810 FOR I=1 TO X-1
820 N=N+1:NL=NL+1
830 IF N=19 THEN N=0:GOSUB 900
840 PRINT"■DATA"N(I)","D(I)","I(I)
860 NEXT I
865 PRINT"■DATA -1 , -1 , -1"
870 GOSUB 900:GOTO 1000
900 PRINT"▓    PRESS ANY KEY"
910 POKE 198,0
920 GET A$:IF A$="" THEN 920
930 PRINT"▓":RETURN
1000 REM
1003 REM ****** END OF PROGRAM ******
1005 REM
```

```
1010 PRINT"■■■■■DO YOU WANT TO DO"
1020 PRINT"■ANOTHER TUNE"
1030 PRINT"■■     (Y OR N)?"
1040 POKE 198,0
1050 GET A$:IF A$<>"Y"AND A$<>"N" THEN 1050
1060 IF A$="Y" THEN RUN
1070 SYS 64802
9000 REM
9003 REM ****** PRESS ANY KEY ******
9005 REM
9010 POKE 198,0:POKE 36878,15
9020 PRINT"■■    PRESS ANY KEY      ■";
9030 GET A$:IF A$="" THEN 9030
9040 POKE 36876,220
9050 FOR I=1 TO 100:NEXT I
9060 POKE 36876,0
9070 RETURN
9100 REM
9103 REM ****** INVALID INPUT ******
9105 REM
9107 FOR J=1 TO 5
9110 PRINT"■■■■■■■■■■■■■■   INVALID INPUT!";
9120 FOR I=225 TO 245
9130 POKE 36877,I
9140 NEXT I:POKE 36877,0
9150 PRINT"■■■■■■■■■■■■■■                 ";
9155 FOR R=1 TO 50:NEXT R,J
9160 RETURN
40000 REM
40003 REM ****** DATA FOR NOTES ******
40005 REM
40010 DATA A,191,B,195,C,201,D,207
40020 DATA E,209,F,215,G,219,H,223
40030 DATA I,225,J,228,K,231,L,232
40040 DATA M,235,N,237,O,239

READY.
```

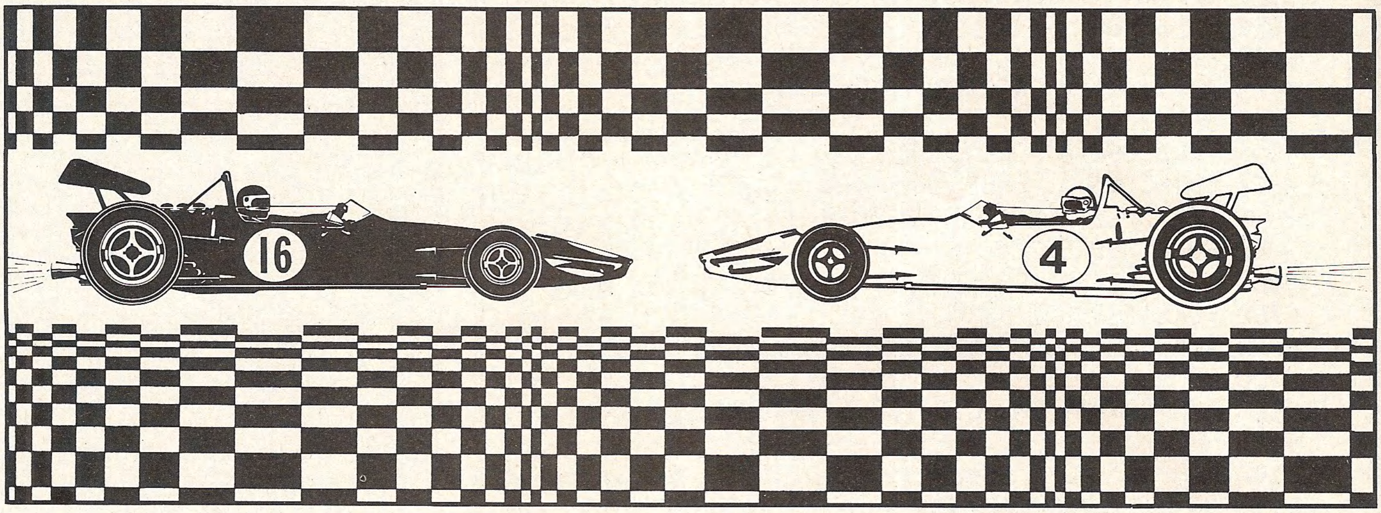




R.H. BOYD

Collision Ahead!

A white racing car heads towards you at breakneck speed! Can you react in time to save yourself?



Get in gear and under starter's orders for an exciting race. You are the driver of a racing car lapping the circuit, eating up the white dots as you speed round in top gear.

DANGER!

But keep alert . . . A white car is completing circuits in the opposite direction to you and you must avoid collisions. Seems easy at first but the game gets trickier once you've eaten the dots.

You move onto the next screen and the hazards get worse as the white car speeds up. It now races towards you at an incredible speed and you must sharpen your reflexes to stay alive. Thinking and action becoming instantaneous. Ready to go?

HOW IT RUNS

10-20	*instructions*
21-148	*set up screen*
150-212	*main program*
215-231	*check if screen cleared*
1000-1020	*print score and men left*
1050-1100	*running score counter*
1200-1280	*set up user defined graphics*

VARIABLES USED

v	*volume*
s1,s3	*noise registers*
H	*score*
W	*men left*
L	*position of cars*
CV	*amount added to POKE colour*

```
4 REM[CLR]
5 PRINT"▒":POKE45,212:POKE46,25:POKE47,212:POKE48,25:POKE49,212:POKE50,25:POKE51
,0:POKE52,28
6 POKE55,0:POKE56,28:POKE36869,255:DIMOL(2),D(2),C(2),CL(2),A(3):V=36878:S1=V-4
8 GOSUB1225
10 IFH>X%THENX%=H
11 POKEV+1,27:PRINT"▒▒▒SCORE:"H"▒REC:"X%:H=0
12 POKE36869,240
15 PRINTTAB(5)"▒▒▒▒▒RACE CAR":PRINTTAB(2)"▒****************"
16 PRINT"▒▒▒▒":PRINT"▒J TO MOVE INWARDS"
17 PRINT"K TO MOVE OUTWARDS":PRINT"A TO ACCELERATE":PRINT"D TO SLOW DOWN"
18 PRINT"▒▒▒▒▒HIT ANY KEY TO START":PRINT"▒▒▒HIT N IF YOU DO NOT   WANT ANOTHER GO
."
19 GETA$:IFA$=""THEN19
20 IFA$="N"THENSYS64802
21 POKE36869,255
100 B$="←←←←←←←←←":C$="#$%&":L$="▒▒▒▒▒▒▒▒▒▒▒▒":D$="▒▒↑▒▒▒↑▒":S3=V-1
110 CV=30720:DO=46:AS=42:O=1:T=22:MO=-1:MT=-22:A(1)=75:A(3)=74:C=3:R=1:Z7=10:Z1=
72:Y=3
```

```
120 PRINT"⅃";:POKEV+1,40:POKEV,0:A$="↑▮▌↑▮▌↑▮▌↑▮▌":P=1:FORI=12TO4STEP-2:GOSUB1000:NE
XT:G=0
130 FORK=1TO2:PRINT"↑"SPC(7)"↑▮██▌↑"SPC(7)"↑";:NEXTK:PRINT"↑███⊠↑█▌";
140 PRINT:P=3:FORI=4TO12STEP2:GOSUB1000:NEXT
142 W=7703:FORI=7TO1STEP-2:FORJ=WTOW+ISTEP2:POKEJ,46:POKEJ+I+6,46:NEXT:W=W+46:NE
XTI
144 W=7703:FORI=8TO2STEP-2:FORJ=WTOW+I*22STEP44:POKEJ,46:POKEJ+2*I+3,46:NEXT:W=W
+46:NEXT
146 W=8121:FORI=8TO2STEP-2:FORJ=WTOW-I*22+22STEP-44:POKEJ,46:POKEJ+2*I+3,46:NEXT
:W=W-42:NEXTI
148 W=8121:FORI=7TO1STEP-2:FORJ=WTOW+ISTEP2:POKEJ,46:POKEJ+I+6,46:NEXT:W=W-42:NE
XTI:W=2:P=32:GOSUB1100
150 C(O)=40:C(W)=41:L(O)=7964:L(W)=8130:CL(O)=3:CL(W)=1:Z=202:FORK=1TO2:POKEL(K)
+CV,CL(K)
151 POKEL(K),C(K):Q(K)=O:OL(K)=P:NEXT:F=O-(R>W)*W:D(O)=MT:D(W)=MO:S=O:M=-(R>W):I
FC<1THEN10
152 POKEV,Z7:FORK=MTOF:J=(INT(K/W)=K/W)+2:POKES1,Z:L=L(J):D=D(J):IFJ=OTHEN160
155 Q=SGN(Q(O)-Q(W))+W:A=A(Q):ONQGOTO190,200,190
160 GETA$:A$=A$+" ":A=ASC(A$):IFA=65THENF=W-(R>Y)*W:M=0:Z=208
162 IFA=68THENF=O-(R>W)*W:M=-(R>W):Z=202
180 IFA<>A(O)ANDA<>A(Y)THEN200
190 TL=L+((D=T)-(D=MT)-(D=O)*T+(D=MO)*T)*((J=W)-(J=O))*((A=A(Y))-(A=A(O))):NL=W*
TL-L
191 N=PEEK(NL):IFPEEK(TL)<>PORN<>POR(PEEK(L+D)<>PANDPEEK(L+D)<>DO)THEN200
192 NL=NL+D:N=PEEK(NL):Q(J)=Q(J)-O*((A=A(Y))-(A=A(O))):ON(N=DO)+WGOTO201,215
200 NL=L+D:N=PEEK(NL):IFN=PTHEN215
201 IFN<>DOTHEN208
202 IFJ=OTHENH=H+5-(R>3)*5:GOSUB1050:POKES1,250:GOTO215
203 IFG>60ANDRND(0)>.5ANDR>1THENN=218
204 GOTO215
208 IFN=218THENPOKES1,240:H=H+25:GOSUB1050:GOTO215
209 IFN=C(Y-J)THEN215
212 D(J)=((D=O)*T+(D=MT)-(D=MO)*T-(D=T))*((J=W)-(J=O)):D=D(J):C(J)=81-C(J):GOTO2
00
215 IFN=C(Y-J)THENGOSUB1200:GOTO150
216 POKEL,OL(J):OL(J)=N:L(J)=NL:POKEL(J),C(J):POKEL(J)+CV,CL(J):NEXTK:IFG<Z1THEN
152
230 POKEV,15:PRINT"⅃BONUS":H=H+100:GOSUB1100:FORI=128TO255:POKES1,I:I=I:NEXT:R=R
+1
231 FORI=1TO99:POKES1,240:POKES1,220:NEXT:POKEV,0:GOTO120
1000 P1=(P-1)*2+1:PRINT"╖"LEFT$(A$,12-I)MID$(C$,P,1)MID$(D$,P1,4)LEFT$(B$,I-4)"
    ";
1005 IFI=4ORI=12THENPRINT"████⊢←←←";
1010 PRINTLEFT$(B$,I-4)MID$(C$,P+1,1);:IFI=12ANDP=3THENRETURN
1015 PRINTMID$(D$,P1,4);:IFI=12THENPRINT"⊠";:RETURN
1020 PRINT"▮▌"LEFT$(A$,12-I):PRINT"⅂"LEFT$(A$,12-I):PRINTTAB(14)"⅂"SPC(I-3)LEFT$(
A$,12-I)"▮▌";:RETURN
1050 G=G+O:N=P
1100 PRINTL$"▀"SPC(10)"+,-":PRINTL$"⊠"SPC(9)MID$(STR$(H),W):PRINTSPC(9)"("STR$(C
):RETURN
1200 POKEL(W),P:POKEL(O),AS:POKES3,Z:FORI=15TO0STEP-.1:POKEV,I:POKEL(O),AS
1220 POKEL(O),P:NEXT:POKEL(O),OL(O):POKEL(W),OL(W):POKES3,0:C=C-1:GOSUB1100:RETU
RN
1225 FORK=1TO13:READX:FORI=0TO7:READA:POKEX+I,A:NEXT:NEXT
1230 FORI=7552TO7631:POKEI,PEEK(I+25600):NEXT
1235 FORI=7176TO7383:POKEI,PEEK(I+25600):NEXT:RETURN
1260 DATA7488,90,90,90,24,24,90,90,90
1262 DATA7496,0,231,66,255,255,66,231,0
1264 DATA7408,8,8,8,8,8,8,8,8
1266 DATA7416,0,0,0,0,255,0,0,0
1268 DATA7448,0,0,0,0,15,8,8,8
1270 DATA7456,0,0,0,0,248,8,8,8
1272 DATA7472,8,8,8,8,248,0,0,0
1274 DATA7464,8,8,8,8,15,0,0,0
1276 DATA7520,238,170,174,172,234,0,0,0
1278 DATA7528,224,128,224,128,224,0,0,0
1282 DATA7512,238,136,232,40,238,0,0,0
1284 DATA7536,0,0,0,0,0,24,24,0
1286 DATA7424,0,0,0,0,0,0,0,0
```

☐

ALLEN WEBB

VIC Enhancers

Larger screen sizes and hi-res graphics for you.

1. GRAPHICS

One of the main criticisms levelled at the VIC-20 is the limitation of the screen size and the need for extra software for high resolution graphics.

Many VIC owners will know how simple it is to improve matters by using user-defined characters. There are, however, additional techniques by which the programmer can improve the graphics and give his program a little more class.

First the bad news. VIC owners with less than 8K memory expansion will not be able to use these techniques properly.

RAISE BASIC

Before giving any details of these techniques, I must first discuss how the start of BASIC can be raised. This will probably be unnecessary to many owners, but please be patient since it is germane to the use of the routines described.

The address of the start of BASIC is held in locations 43 and 44 as two bytes. These values must be reset to the required values. In addition, the byte before the start of BASIC must be zero. If the required start of BASIC is SA, the following line in command mode will carry out the required action:

```
POKE 44,SA/256: POKE 43,SA-
PEEK(44)*256: POKE SA-1,0: NEW
```

Don't worry if you make a mistake in the typing. Any incorrect values will simply result in a SYNTAX ERROR message and will do no damage.

SCREEN SIZE

Most of what comes next is covered in the CBM VIC-20 Programmer's Reference Guide. I fully recommend this book for further reading.

As most critics are willing to tell us, the normal size of the VIC screen is 23 rows of 22 characters. The actual number of rows is controlled by bits 1-6 of location 36867. Any attempts to increase the number of rows will give extra rows of odd characters at the bottom of the screen.

Any attempt to move the cursor to this new area is foiled since the operating system does not recognise this extra screen. This area is, however, contiguous with the normal screen and can be poked with characters. The colour array behaves in the same way. Listing 1 gives a routine for the generation of a screen containing 30 rows of 22 characters.

Since the screen occupies a greater area than normal, and to provide space for user defined characters, the bottom of BASIC must be raised to 6200 (decimal) or above. A short machine code routine is located at 5000 (decimal) to clear the new screen. Line 190 provides a set of standard characters in RAM and the first poke in line 110 sets the pointers to the user defined character area. For operation without user-defined characters, delete line 190 and the first poke in line 110.

CURSOR SHIFT

Listing 2 gives a short routine which will fool the operating system into placing the print cursor at any specified row and column on the extended screen. A print statement executed after this routine will be printed at the required position. Any print statements should finish with a semi-colon to suppress scrolling of the normal screen area.

The expanded screen offers greater flexibility for programs using mainly text, like word processing and adventures, as well as providing a little more space for arcade-type games.

EXTRA SCREENS

Most owners will be aware that during expansion, the screen memory moves from 7680 (decimal) to 4096 (decimal). This is all to do with efficient use of memory. The VIC will, however, allow up to six screen locations.

The screen position is controlled by bit 7 of location 36866 and bits 4-7 of location 36869. Location 648 contains the most significant byte of the screen start address.

The position of the screen can be changed by two methods.

- In direct mode...
  POKE 648, Z followed by RETURN and RUN/STOP+RESTORE.

- In a program...
  10 POKE 36866,X: POKE 36869,Y: POKE 648,Z

The values for X, Y and Z are given in table 1. In order to use all of these screens, the bottom of BASIC should be raised to 8192 (decimal). Since each screen can be used independently, simple animation effects can be demonstrated. Listing 3 gives a demonstration routine. Lines 120 to 170 set up each screen in turn by performing the required pokes. Lines 180 to 230 recall each screen in sequence. Line 210 is a simple delay and can be removed to demonstrate the effects at full speed.

To use user defined characters with a relocated screen, a fair amount of bit twiddling is necessary. Listing 4 is a short program which will provide all necessary data for all screen-character set configurations.

The only way to use all of the multiple screens with user defined characters would probably be to locate the character set at 5120 (decimal). I haven't had a chance to check that yet.

UDCs

User defined characters are of great value but are not really any use for direct uses such as graph plotting, sketching, drawing a man in an adventure and so on. If you have enough spare cash, a Super Expander will be a good investment. If not, and you don't mind the slow speed of BASIC, Listing 5 might be of value.

This routine must be used with the bottom of BASIC at 8300 (decimal). A machine code routine at 8200 (decimal) sets up the screen to a 176 by 160 resolution. The routine does not trap out of range values for X and Y. You must do this.

These routines are, not unexpectedly, of a general nature. They are intended to offer useful alternatives to VIC owners and, I hope, provoke a little interest. Due to the limitations of space, it is not feasible to describe in detail how each routine works. The text gives an outline description of each. I suggest you experiment with the routines and read the book recommended earlier.

2. BASIC

One of the main failings of Commodore is their inability to extend the resident BASIC to allow ease of use of the facilities offered by their machines.

For VIC owners, a Super Expander fills some of the gaps in the BASIC, but at a price. After nearly two years of using a VIC, I realised that it was time that I rationalised the use of some of the VIC's facilities. This article is the result.

One of the functions given in the routine works only on a VIC with eight or more kilobytes expansion and bearing in mind the lack of RAM in the basic machine, the routine is only available on machines with at least 8K expansion.

All commands are accessed by a single SYS call (SYS 24128(16K expansion) or SYS 15936(8K expansion)). The command is defined by two letters followed by any parameter: SYS (SA) command, parameter 1, parameter 2.

The actual syntax has a little flexibility in that spaces can be included to ease reading. The parameters can be numeric values, variables or functions. The one limitation is that if you use a variable for the SYS address, then brackets must be used. Here are some examples of acceptable syntax: SYS24128PA,1,1; SYS(24128) PA, RND(1)*10,Y12; but SYS SA PA,1,1 will give a syntax error.

Specifically, the commands available are:

Colours: SYS (SA) BC, colour sets the border colour; SYS (SA) SC, colour sets the screen colour.

Sound: SYS(SA) VO, volume sets volume; SYS (SA) TU, voice, frequency sets tone.

Print at: SYS (SA) PA, X, Y sets the cursor to the specified coordinates.

Line erase: SYS (SA) LE, Y erases specified line.

Set height of screen: SYS (SA) RO, number of rows sets up required number of rows.

Set top of memory: SYS(SA)HI, address of top of memory. For example, to set the top of memory to $5000 use SYS (SA) HI,5*4096

Change location of screen in RAM: SYS (SA) FR, screen number (in the range 1 to 8). The VIC with more than 8K expansion supports up to eight screens between $1000 and $2000, as I discussed above. This command will switch to the screen position specified in the parameter. A parameter value out of the range 1 to 8 will result in a nasty crash. Before you can access all of the screens you must raise the start of BASIC with: POKE 642,32:SYS 64824.

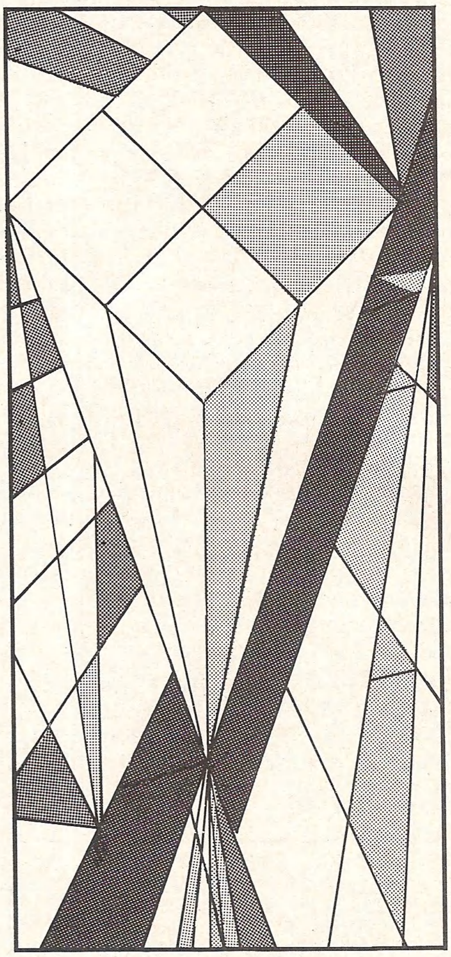
Table 1

Screen position (decimal)	*Colour Array position (decimal)*	*X*	*Y*	*Z*
4096	*37888*	*22*	*192*	*16*
4608	*38400*	*150*	*192*	*18*
6144	*37888*	*22*	*224*	*24*
6656	*38400*	*150*	*224*	*26*
7168	*37888*	*22*	*240*	*28*
7680	*38400*	*150*	*240*	*30*

Listings 1 and **2** gives the versions for 8K and 16K. **Listing 3** gives a short demo which should clarify the use of some of the commands (don't forget to raise the start of BASIC before running it!)

```
0 REM   LISTING 1
1 DATA32,121,0,201,66,208,3,76,131,94,201
2 DATA83,208,3,76,166,94,201,80,208,3,76,206
3 DATA94,201,76,208,3,76,254,94,201,82,208
4 DATA3,76,23,95,201,84,208,3,76,58,95,201
5 DATA86,208,3,76,108,95,201,70,208,3,76,142
6 DATA95,201,72,208,3,76,220,95,96,32,115
7 DATA0,201,67,208,25,32,115,0,32,253,206
8 DATA32,138,205,32,247,215,198,20,173,15
9 DATA144,41,248,5,20,141,15,144,96,76,8,207
10 DATA32,115,0,201,67,208,246,32,115,0,32
11 DATA253,206,32,138,205,32,247,215,198,20
12 DATA165,20,10,10,10,10,133,1,173,15,144
13 DATA41,15,5,1,141,15,144,96,32,115,0,201
14 DATA65,208,206,32,115,0,32,253,206,32,138
15 DATA205,32,247,215,165,20,133,1,32,121
16 DATA0,32,253,206,32,138,205,32,247,215
17 DATA165,20,133,2,166,1,164,2,24,32,240
18 DATA255,96,32,115,0,201,69,208,158,32,115
19 DATA0,32,253,206,32,138,205,32,247,215
20 DATA166,20,32,141,234,96,32,115,0,201,79
21 DATA208,133,32,115,0,32,253,206,32,138
22 DATA205,32,247,215,165,20,10,133,20,173
23 DATA3,144,41,129,5,20,141,3,144,96,32,115
24 DATA0,201,85,240,3,76,163,94,32,115,0,32
25 DATA253,206,32,138,205,32,247,215,165,20
26 DATA133,1,32,121,0,32,253,206,32,138,205
27 DATA32,247,215,165,20,133,2,166,1,165,2
28 DATA157,9,144,96,32,115,0,201,79,240,3
29 DATA76,163,94,32,115,0,32,253,206,32,138
30 DATA205,32,247,215,165,20,201,15,48,2,169
31 DATA15,141,14,144,96,32,115,0,201,82,240
32 DATA3,76,163,94,32,115,0,32,253,206,32
33 DATA138,205,32,247,215,166,20,224,8,48
34 DATA2,162,8,133,1,189,193,95,141,2,144
35 DATA189,202,95,141,5,144,189,211,95,141
36 DATA136,2,96,0,22,150,22,150,22,150,22
37 DATA150,0,192,192,208,208,224,224,240,240
38 DATA0,16,18,20,22,24,26,28,30,32,115,0
39 DATA201,73,240,3,76,163,94,32,115,0,32
40 DATA253,206,32,138,205,32,247,215,165,20
41 DATA133,55,133,51,165,21,133,56,133,52
42 DATA96
43 FORI=24128TO24574
44 READX:POKEI,X
45 NEXT
46 POKE55,57:POKE51,57
47 POKE56,94:POKE52,94
50 REM
60 REM ENHANCED BASIC 16K VERSION A E WEBB 1983
70 REM
```

```
0 REM   LISTING 2
1 DATA32,121,0,201,66,208,3,76,131,62,201
2 DATA83,208,3,76,166,62,201,80,208,3,76,206
3 DATA62,201,76,208,3,76,254,62,201,82,208
4 DATA3,76,23,63,201,84,208,3,76,58,63,201
5 DATA86,208,3,76,108,63,201,70,208,3,76,142
6 DATA63,201,72,208,3,76,220,63,96,32,115
7 DATA0,201,67,208,25,32,115,0,32,253,206
8 DATA32,138,205,32,247,215,198,20,173,15
9 DATA144,41,248,5,20,141,15,144,96,76,8,207
10 DATA32,115,0,201,67,208,246,32,115,0,32
11 DATA253,206,32,138,205,32,247,215,198,20
12 DATA165,20,10,10,10,10,133,1,173,15,144
13 DATA41,15,5,1,141,15,144,96,32,115,0,201
14 DATA65,208,206,32,115,0,32,253,206,32,138
15 DATA205,32,247,215,165,20,133,1,32,121
16 DATA0,32,253,206,32,138,205,32,247,215
17 DATA165,20,133,2,166,1,164,2,24,32,240
18 DATA255,96,32,115,0,201,69,208,158,32,115
19 DATA0,32,253,206,32,138,205,32,247,215
20 DATA166,20,32,141,234,96,32,115,0,201,79
21 DATA208,133,32,115,0,32,253,206,32,138
22 DATA205,32,247,215,165,20,10,133,20,173
23 DATA3,144,41,129,5,20,141,3,144,96,32,115
24 DATA0,201;85,240,3,76,163,62,32,115,0,32
25 DATA253,206,32,138,205,32,247,215,165,20
26 DATA133,1,32,121,0,32,253,206,32,138,205
27 DATA32,247,215,165,20,133,2,166,1,165,2
28 DATA157,9,144,96,32,115,0,201,79,240,3
29 DATA76,163,62,32,115,0,32,253,206,32,138
30 DATA205,32,247,215,165,20,201,15,48,2,169
31 DATA15,141,14,144,96,32,115,0,201,82,240
32 DATA3,76,163,62,32,115,0,32,253,206,32
33 DATA138,205,32,247,215,166,20,224,8,48
34 DATA2,162;8,133,1,189,193,63,141,2,144
35 DATA189,202,63,141,5,144,189,211,63,141
36 DATA136,2,96,0,22,150,22,150,22,150,22
37 DATA150,0,192,192,208,208,224,224,240,240
38 DATA0,16,18,20,22,24,26,28,30,32,115,0
39 DATA201,73,240,3,76,163,62,32,115,0,32
40 DATA253,206,32,138,205,32,247,215,165,20
41 DATA133,55,133,51,165,21,133,56,133,52
42 DATA96
43 FORI=15936TO16382
44 READX:POKEI,X
45 NEXT
46 POKE56,62:POKE52,62
47 POKE55,64:POKE51,64
50 REM
60 REM ENHANCED BASIC 8K VERSION A E WEBB 1983
70 REM
```



```
0 REM   LISTING 3
10 REM
20 REM ENHANCED BASIC DEMO
30 REM
40 SA=24128
50 FORI=1TO8
60 SYS(SA)FR,I
70 PRINTCHR$(147)
80 FORJ=1TO100
90 SYS(SA)PA,RND(1)*20,RND(1)*20:PRINT"*"
100 NEXT J
110 NEXT I
120 SYS(SA)VO,15
130 FORI=1TO8
140 SYS(SA)SC,RND(1)*15+1
150 SYS(SA)BC,RND(1)*8+1
160 SYS(SA)TU,1,RND(1)*50+200
170 SYS(SA)FR,I
180 SYS(SA)RO,RND(1)*10+10
190 NEXT I
200 GOTO 130
```

Listing 4

```
1 REM ALLEN WEBB 1982
10 DATA 4096,252,5120,253,6144,254,7168,255
20 DATA 4096,16,192,4608,18,192,6144,24,224,6656,26,224,7168,28,240,7680,30,240
30 FORL=1TO4
40 READCP(L),P1(L):NEXT
50 REM CP=CHAR POS P1=36869 POKE
60 FORL=1TO6
70 READ SP(L),P2(L),P3(L):NEXT
80 REM SP=SCREEN POS P2=648 POKE P3,36869 POKE
90 PRINT"□   VIC CONFIGURATION          ROUTINE"
100 PRINT"──────────────────────"
110 PRINT"▣YOU MUST HAVE AT LEAST8K EXPANSION AND PUT  THE START";
120 PRINT" OF BASIC TO $2000 TO USE ALL OF   THE COMBINATIONS OF   ADDRESSES GI
VEN."
130 PRINT"▣▣▣ ▣PRESS SPACE FOR MORE"
140 GETI$:IFI$<>" "THEN140
150 PRINT"□    SCREEN LOCATION"
160 PRINT"──────────────────────"
170 PRINT"▣THE SCREEN CAN BE PUT AT THE FOLLOWING      LOCATIONS."
180 PRINT"▣"
190 FORL=1TO6:O$=STR$(L)+".  $"+RIGHT$(STR$(SP(L)),4)
200 PRINTO$:NEXT
210 PRINT"▣▣PLEASE SELECT"
220 GETI:IFI<1ORI>6THEN220
230 SP=SP(I):PA=P2(I):PB=P3(I)
240 PC=22:CA=37888:IFINT(I/2)=I/2THENPC=150:CA=38400
250 PRINT"□THE SCREEN CAN BE PUT AT $";SP;"BY:"
260 PRINT"▣1. POKE648,"PA;"IN      DIRECT MODE. FOLLOWED BY RESTORE."
270 PRINT"▣OR":PRINT"2. POKE648,";PA;"             POKE36869,"PB,"   POKE36866,"PC
280 PRINT"IN A PROGRAM."
290 PRINT"▣THE COLOUR ARRAY WILL START AT";CA
300 PRINT"▣▣▣ ▣PRESS SPACE FOR MORE"
310 GETI$:IFI$<>" "THEN310
320 PRINT"□CHARACTER SET LOCATION"
330 PRINT"□──────────────────────"
340 PRINT"THE CHARACTER SET CAN BE PUT AT:"
350 PRINT"▣"
360 FORL=1TO4:O$=STR$(L)+".  $"+RIGHT$(STR$(CP(L)),4)
370 PRINTO$:NEXT
380 PRINT"▣▣PLEASE SELECT"
390 GETI:IFI<1ORI>4THEN390
400 CP=CP(I):PD=P1(I)
410 IFCP<>SPTHEN470
420 PRINT"▣THE CHARACTER SET AND SCREEN CANNOT BE AT   THE SAME ADDRESS"
430 PRINT"▣DO YOU WANT TO SELECT ANOTHER ADDRESS (Y/N)?"
440 GETI$:IFI$=""THEN440
450 IFI$="Y"THEN340
460 END
470 FP=(PBAND240)OR(PDAND15)
480 PRINT"▣POKE36869,"FP;"WILL   SET THE SCREEN AND    CHARACTER SET TO THE  RE
QUIRED";
490 PRINT" ADDRESSES."
```

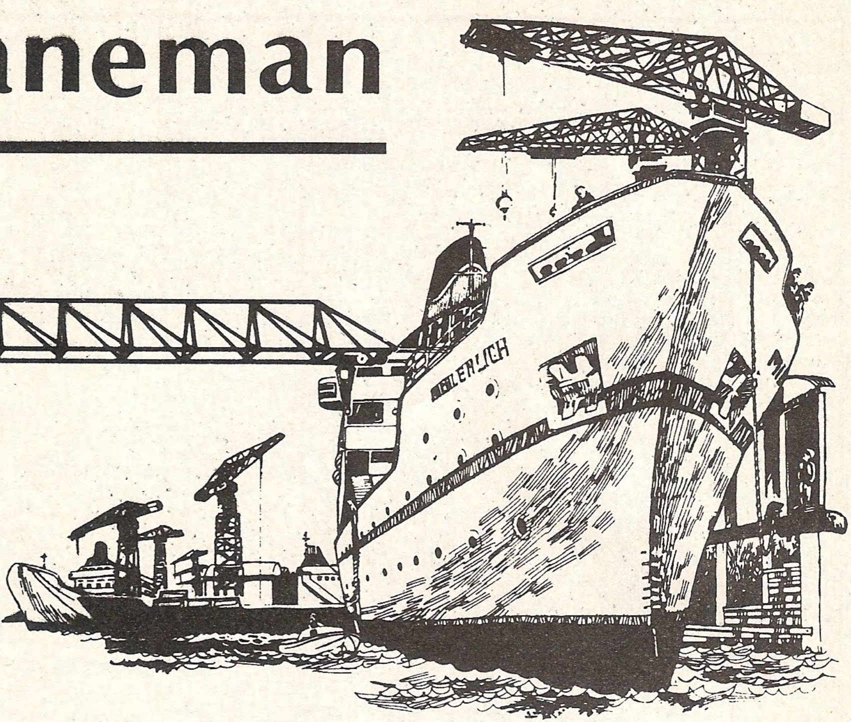
Listing 5

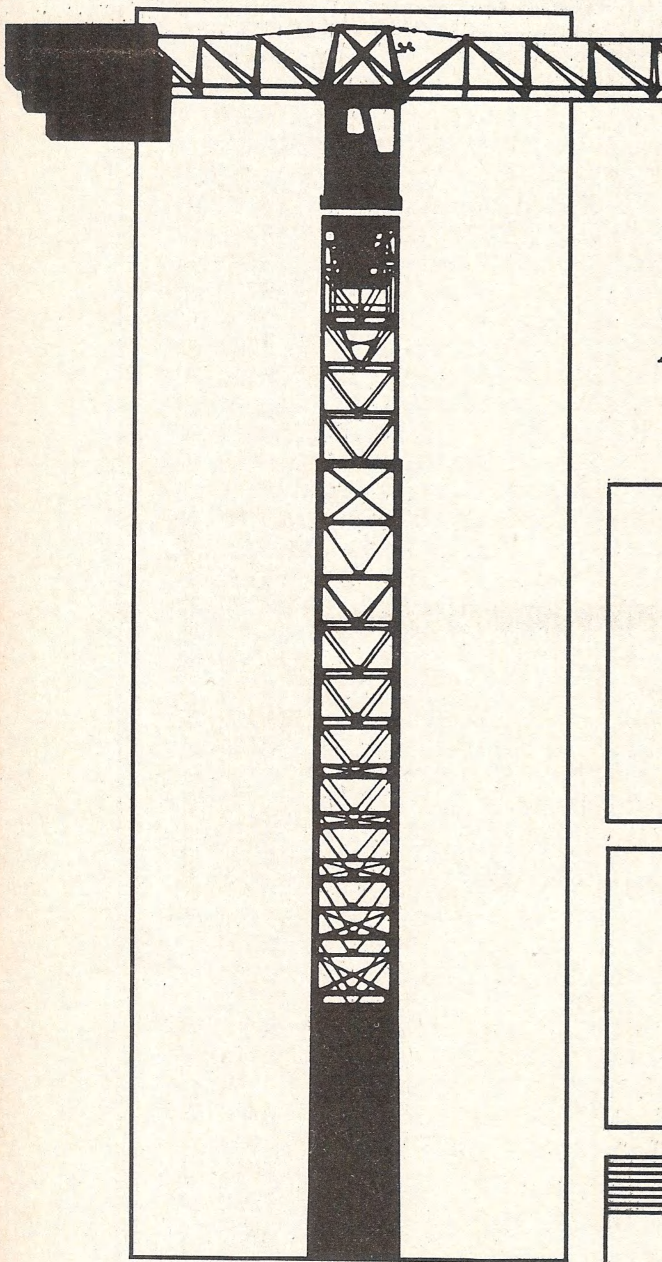
```
10 IFPEEK(8200)=72ANDPEEK(8210)=169ANDPEEK(8284)
   =96THEN110
20 DATA72,152,72,138,72,169,252,141,5,144
30 DATA169,21,141,3,144,169,0,133,87,170,165
40 DATA87,157,0,30,169,6,157,0,150,230
50 DATA87,232,224,220,208,239,169,0,133,87
60 DATA169,16,133,88,162,0,160,0,169,0
70 DATA145,87,200,192,22,208,247,232,224,160
80 DATA240,16,24,165,87,105,22,133,87,165
90 DATA88,105,0,133,88,76,55,32,104,170,104,168,104,96
100 FORL=8200TO8284:READX:POKEL,X:NEXT
110 SYS8200
120 GOTO1000
130 C=INT(Y/16)*22+INT(X/8)
140 R=(Y/16-INT(Y/16))*16
150 B1=4096+C*16+R
160 B2=7-(X-INT(X/8)*8)
170 IFSF=1THENPOKEB1,PEEK(B1)OR(2↑B2)
    :POKE38400+C,CO:RETURN
180 IFSF=0THENPOKEB1,PEEK(B1)AND(255-2↑B2):RETURN
1000 REM***************************
1010 REM MAIN PROGRAM STARTS HERE
1020 REM***************************
1030 REM
1040 REM***************************
1050 REM    HI-RES PLOT ROUTINE
1060 REM     X=0-175   Y=0-159
1070 REM      (A E WEBB 1982)
1080 REM
1090 REM MEMORY MAP...
1100 REM 4096-7615..CHARACTER SET
1110 REM 7680-8186..SCREEN
1120 REM 8200-8224..MACHINE CODE
1130 REM 8300.......START OF BASIC
1140 REM
1150 REM***************************
```

ANDREW McNALLY

Crazy Craneman

Make sure you don't get fired in this test of skill.





You have to load a ship with a crane in this game for the unexpanded VIC-20. On screen is a crane on a quayside with you in charge of its arm.

At the dock beneath you there is a ship waiting to be loaded but, to make things interesting, it moves backwards and forwards. Your job is to press a key at the right time to lower the cable, pick up the crates and deposit them on the ship.

After you have loaded crates successfully the crane becomes higher, making your job harder.

HOW IT RUNS

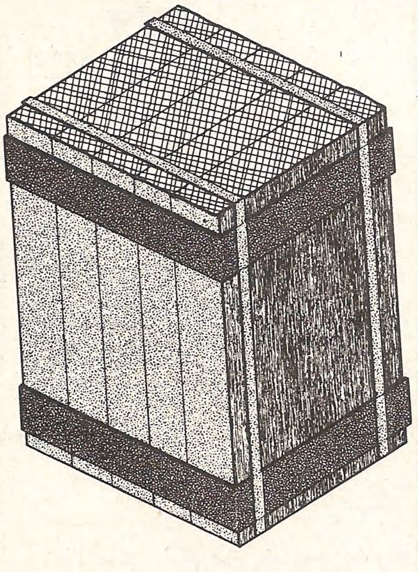
5-97	*set up screen and variables*
100-112	*print crane and ship*
150	*check key depression*
160-217	*change direction of ship and hook*
1000-1830	*lower line, deposit or collect crates*
5000-5100	*raise crane*
8000-8040	*end game*
9000-9050	*end game when ship loaded*
10000-10030	*instructions*

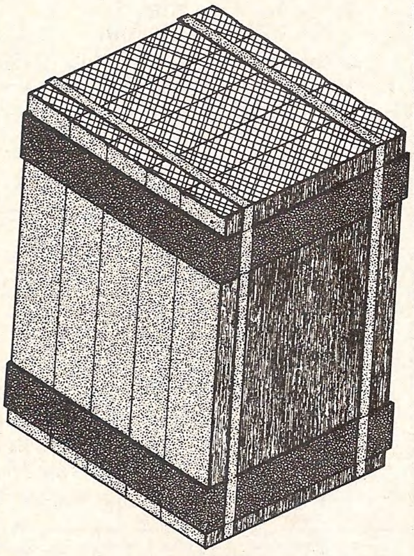
VARIABLES USED

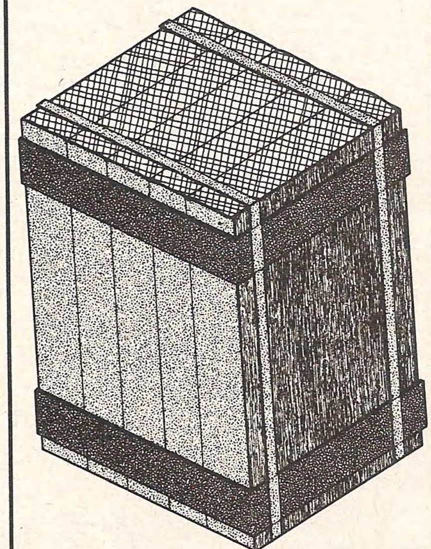
H	*height of crane*
S	*score*
X	*position of ship*
E	*position of crane arm*
K	*direction of ship*
M	*direction of crane arm*
T	*time*

```
1 GOSUB10000
2 H=1
3 POKE36878,15
5 S=0
9 REM [CLR]
10 PRINT"♥":POKE36879,184
11 PRINT"SESCORE="S:REM [HOME] [WHT]
12 FORF=0TO21:POKE8142+F,160:POKE8164+F,160
14 POKE38862+F,6:POKE38884+F,6:NEXT
15 REM [HOME][20 CRSR DWN] [BLK] [RVS ON]
16 PRINT"SQQQQQQQQQQQQQQQQQQQQ▮R    "
17 REM [2 CRSR UP] [RVS ON] [RED]
18 PRINT"]]R  ▚ "
30 REMVARIABLES
31 L=6:X=5:M=1:K=1:E=0:C=75
33 Z=0
50 REMSET UP
51 T=0
60 POKE8164-((H*22)+66),86:POKE8164-((H*22)+66)+30720,5
```

```
68 POKE8164-((H*22)+88),160:POKE8164-((H*22)+88)+30720,4
95 R=0
97 Q=8164-((H*2[illegible])+88)
100 IFR=0THEN:POKEE+Q+1,67:POKEE+Q+1+30720,0
102 IFR=1THENPOKEQ+E+1,160:POKEQ+E+1+30720,2
108 J=8120+X:U=38840+X
110 POKEJ,121:POKEU,0
111 POKEJ-1,95:POKEU-1,0:POKEJ-2,32
112 POKEJ+1,105:POKEU+1,0:POKEJ+2,32
150 IFPEEK(203)<>64THENGOSUB1000
160 T=T+1
162 IFT=50THEN8010
200 E=E+M
201 IFE=19THENM=-1
202 IFE=1THENM=1
205 POKEQ+E,67:POKEQ+E+30720,0:POKEQ+E+2,32
215 X=X+K
216 IFX=19THENK=-1
217 IFX=5THENK=1
300 GOTO100
1000 IFR=1THEN1005
1002 IFR=0THEN1100
1005 GOSUB1800
1015 IFE<>X-1THEN8000
1016 POKE36875,240:FORF=0TO120
1017 NEXT:POKE36875,0
1020 R=0
1021 T=0
1025 S=S+1:Z=Z+1
1026 REM [HOME][WHT]
1027 PRINT"▒▒SCORE="S
1030 IFZ=2THENGOTO5000
1090 GOSUB1820
1099 RETURN
1100 GOSUB1800
1106 IFE<>1THEN8000
1120 POKE8078,32
1121 POKE36875,199:FORF=0TO120
1122 NEXT:POKE36875,0
1125 R=1
1126 T=0
1180 GOSUB1820
1190 RETURN
1800 FORF=Q+22TO8076STEP22
1802 POKEF+E+1,66:POKEF+E+1+30720,0
1804 NEXTF
1810 RETURN
1820 FORF=8076TOQ+22STEP-22
1822 POKEF+E+1,32:NEXT
1830 RETURN
5000 H=H+1
5050 IFH=16THEN9000
5100 GOTO10
6000 REM [HOME][ 5 CRSR DOWN][3CRSR RIGHT][RED]
8005 GOTO8020
8006 REM [HOME][ 6 CRSR DOWN][3CRSR RIGHT][RED]
8010 PRINT"▒▒▒▒▒▒▒▒▒▒▒TOO SLOW !"
8020 POKE36877,220
8021 FORL=15TO0STEP-1
8022 POKE36878,L
8023 FORM=1TO300
8024 NEXTM
8025 NEXTL
8026 POKE36877,0
8027 REM [CRSR DOWN] [PUR]
8030 PRINT"▒▒PRESS [S] TO START"
8035 GETA$
8036 IFA$<>"S"THEN8035
8040 RUN
8506 REM [HOME][ 6 CRSR DOWN][3CRSR RIGHT][RED]
9000 PRINT"▒▒▒▒▒▒▒▒▒▒▒WELL DONE CRANEMAN"
9001 FORB=0TO5
9002 FORF=128TO241
9003 POKE36875,F
9010 NEXTF
9012 NEXTB
9013 POKE36878,0
9014 POKE36875,0
9050 GOTO8030
9999 REM [HOME][ 3 CRSR DOWN][3CRSR RIGHT][RED]
10000 PRINT"▒▒▒▒▒▒▒▒CRAZY CRANEMAN"
10001 PRINT"▒▒▒▒▒LOAD THE SHIP" :REM [3 CRSR DOWN][CRSR RIGHT][BLK]
10002 PRINT"▒BY PRESSING A KEY" :REM [CRSR RIGHT]
10003 PRINT"▒TO LOWER THE LINE" :REM [CRSR RIGHT]
10005 PRINT"▒▒▒▒▒▒BUT BE QUICK OR YOUR  FIRED":REM[4 CRSR DOWN][GRN][RIGHT]
10006 PRINT"▒▒▒▒▒▒▒▒▒______________________":REM[HOME][4CRSR DOWN][3CRSR RIGHT][PUR]
10007 POKE36878,15
10010 FORV=1TO10:READA:READJ:POKE36875,A
10012 FORI=0TOJ:NEXTI:POKE36875,0
10014 NEXTV
10020 DATA229,300,231,300,229,600,231,600
10021 DATA225,300,227,300,225,600,227,600
10022 DATA230,700,233,1000
10030 RETURN
```



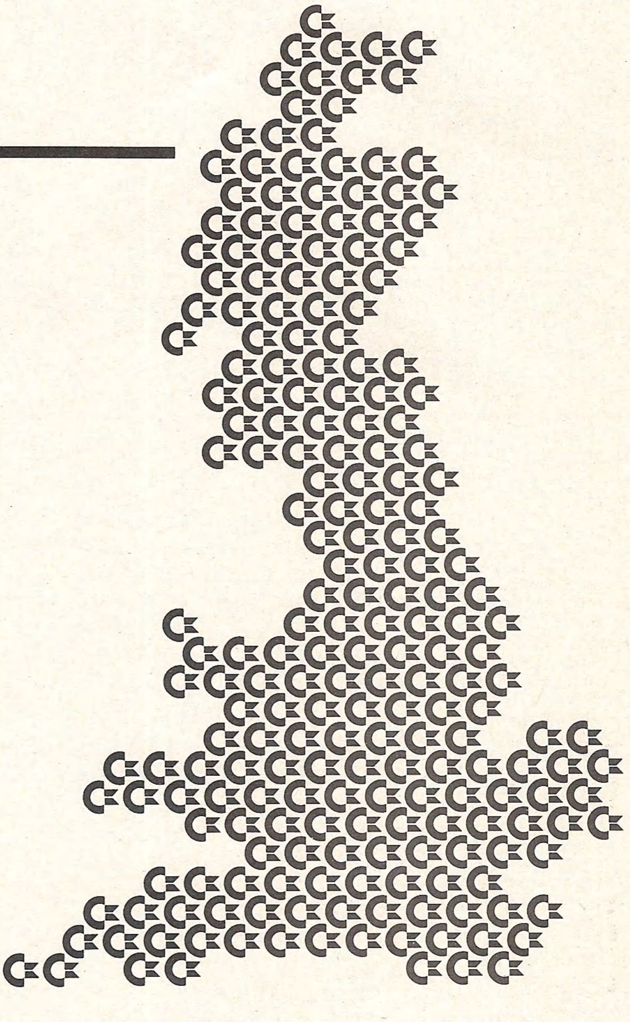




P. STODDART

Great Britain

Draw Great Britain and Eire to learn how to really use the definable characters on the VIC-20.



This short program for the unexpanded VIC-20 will show you a detailed map of Great Britain and Eire.

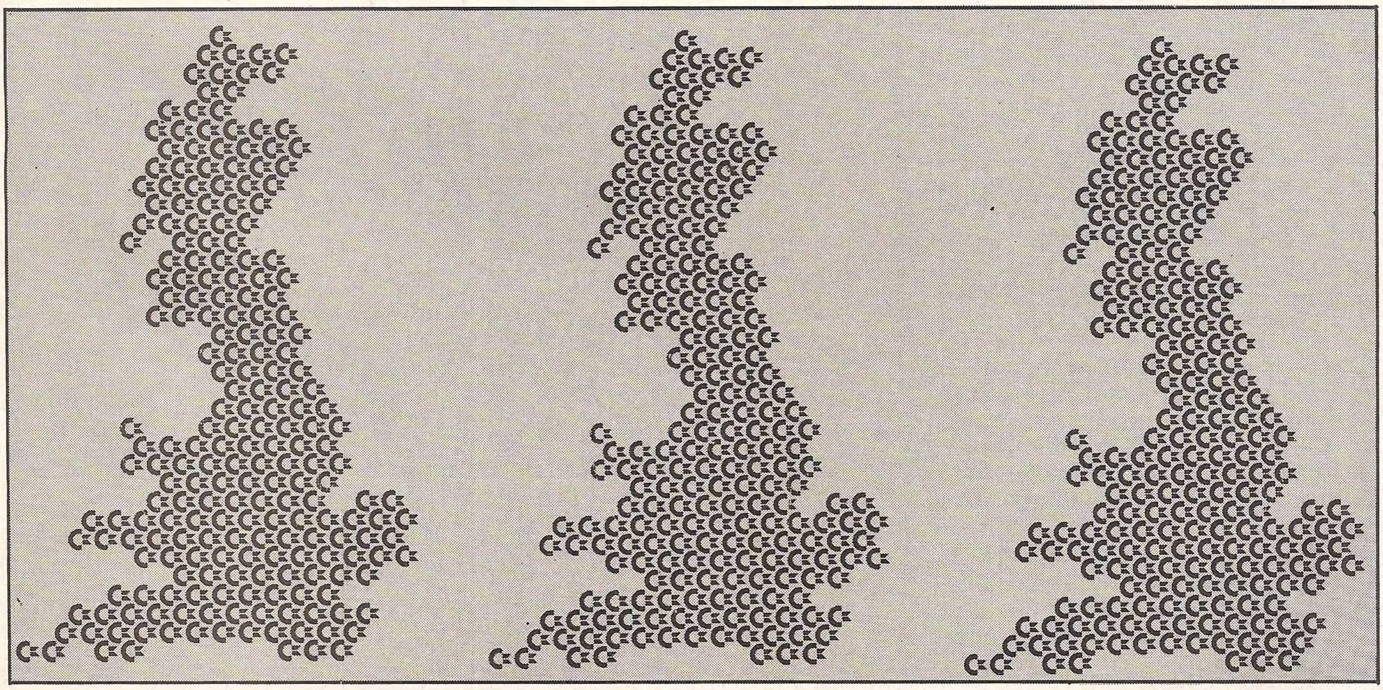
The data lists may look a little daunting but you'll find it worth your trouble to be accurate. There are lessons too to be learnt about the VIC-20's definable characters.

HOW IT RUNS

10	*lowers RAMtop to protect your new characters*
40	*reads the data and POKEs it into the character set*
1000-2000	*changes screen colour and prints the map onto the screen*

```
10 POKE52,28:POKE56,28:CLR
40 FORC=7168TO7671:READA:POKEC,A:NEXT
50 DATA0,3,3,1,1,3,3,7
60 DATA0,131,255,255,255,255,255,255
70 DATA56,248,248,240,240,224,192,0
80 DATA0,0,0,0,119,255,255,255
90 DATA254,252,254,254,252,255,255,255
100 DATA3,15,31,31,15,31,31,7
110 DATA0,0,0,0,0,240,240,224
120 DATA192,192,192,128,128,128,0,0
130 DATA7,15,31,31,127,127,95,31
140 DATA31,15,15,15,31,127,95,31
160 DATA254,224,252,252,240,224,246,255
170 DATA128,224,240,240,248,248,248,248
180 DATA25,49,32,32,32,96,3,3
190 DATA7,7,135,130,66,192,192,192
200 DATA255,255,255,248,64,0,0,0
210 DATA248,252,252,252,254,254,254,255
220 DATA192,240,248,248,252,252,252,247
230 DATA255,255,255,207,15,31,63,63
240 DATA0,0,192,224,207,191,63,255
250 DATA63,127,127,31,31,191,255,255
260 DATA242,248,254,255,255,255,255,255
270 DATA0,0,0,0,0,128,128,0
280 DATA254,254,255,255,255,255,255,255
290 DATA24,63,191,255,255,255,255,255
300 DATA0,0,192,192,224,192,224,192
310 DATA1,3,0,0,0,0,0,0
320 DATA255,63,63,31,31,63,63,127
330 DATA224,224,192,0,0,0,0,0
340 DATA255,255,255,255,255,254,248,248
350 DATA255,255,255,255,255,255,255,207
360 DATA127,255,255,255,255,255,127,111
370 DATA0,7,31,127,31,62,8,0
380 DATA0,0,0,0,0,0,0,0
390 DATA0,0,0,0,0,3,3,3
400 DATA15,7,0,48,127,255,255,255
410 DATA31,63,63,255,255,255,255,255
420 DATA240,224,0,227,255,252,248,240
430 DATA0,0,0,128,128,0,0,0
```

```
440 DATA192,128,0,0,0,0,0,0
450 DATA255,47,0,0,0,0,0,0
460 DATA248,224,142,15,26,0,0,0
470 DATA255,159,7,0,0,0,0,0
480 DATA255,255,255,248,120,48,0,0
490 DATA7,15,63,63,62,240,224,192
500 DATA0,0,0,0,0,0,7,6
510 DATA255,255,255,255,255,255,255,255
520 DATA0,0,0,0,0,128,60,254
530 DATA0,0,0,0,0,7,127,255
540 DATA127,255,255,63,63,255,255,255
550 DATA0,0,0,0,0,1,3,255
560 DATA0,0,0,0,0,0,0,7
570 DATA3,7,0,0,3,7,31,15
580 DATA15,3,0,0,1,1,1,3
590 DATA255,255,63,255,255,255,255,255
600 DATA31,255,255,255,255,255,255,255
610 DATA14,0,15,15,255,31,63,127
620 DATA255,127,95,15,31,1,0,0
630 DATA255,255,255,250,224,0,0,0
640 DATA240,192,128,0,0,0,0,0
650 DATA248,240,240,224,224,224,192,0
660 DATA252,252,252,248,252,252,252,248
670 DATA255,255,254,254,248,248,248,248
680 DATA192,128,0,0,0,0,0,0
1000 POKE36879,8:POKE36869,255
1001 PRINT"█";:PRINT"█---------------------";
1002 PRINT"██GREAT BRITAIN AND EIRE██";:PRINT"----------------------"
1003 PRINT" ███      @AB":GOSUB2000
1004 PRINT"        EDCF":GOSUB2000
1005 PRINT"        H--G":GOSUB2000
1006 PRINT"        I-J":GOSUB2000
1007 PRINT"      /.L--K":GOSUB2000
1008 PRINT"    210-MN-O":GOSUB2000
1009 PRINT"    3--=> Q-P":GOSUB2000
1010 PRINT"    45-< RS-TU":GOSUB2000
1011 PRINT"    76-;YZ--VWX":GOSUB2000
1012 PRINT"    89: ←↑]--\[":GOSUB2000
1013 PRINT"        !"CHR$(34)"#--$%":GOSUB2000
1014 PRINT"       ,+*)(/&
1020 FORT=1TO2000:NEXT:PRINT"████   1984 P.STODDART"
1021 PRINT"██      FOR   H.C.W"
1030 GOTO1030
2000 FORT=1TO75.5:NEXT:RETURN
```



LES ALLEN

The Little Professor

Sophisticated graphics make arithmetic more fun.

The Little Professor program utilises user defined graphics to produce the face of the professor and provide text for the questions being set. Sprite graphics are used to provide the title sequence and text for the mode of question set.

The menu at the start allows the pupils the choice of 10 questions from one of four function and nine skill levels. Three attempts at each question are allowed before the correct answer is given.

After completion of the 10 questions, the menu gives the following options: further tests on same function (key f1); change function and/or skill level (f3); end of test (f7).

An incorrect entry can be deleted by pressing DEL and the solution to the sum is made by pressing RETURN. As a hint to the solution, the length of line under the lower figure is adjusted for each question.

The program is loaded in two parts:

DATA FILE

This provides the basic 'boot' for the user defined graphics, sprite graphics and machine code routines which are temporarily SAVEd as a file at decimal address 32768.

The program must be entered and SAVEd prior to running and extreme care taken to ensure that the DATA is correct as a missed comma can make all the difference. Error trapping routines are provided to assist in locating the error to the relevant line number.

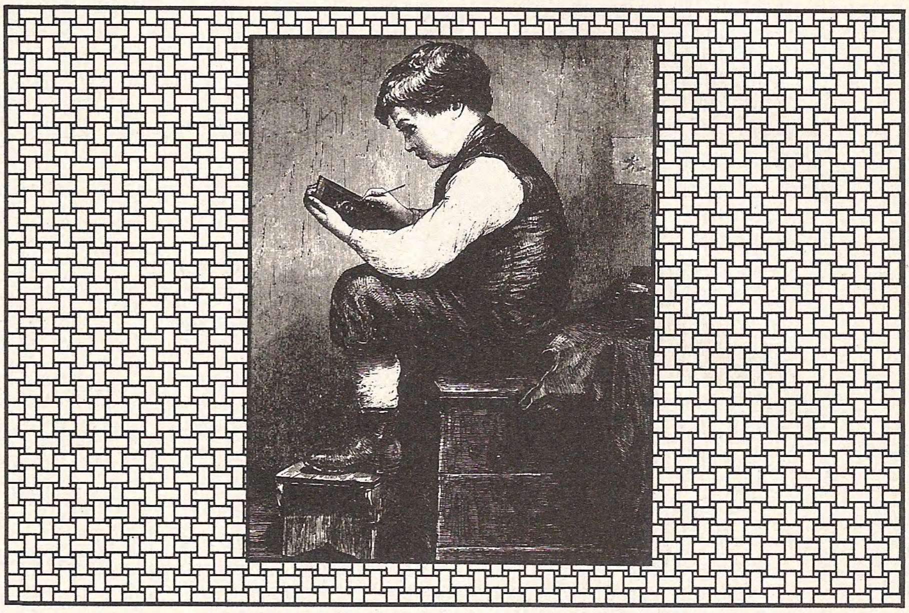
When the program has been correctly entered it must be SAVEd prior to viewing the effects of the graphics — when prompted press N to view the screen. Warning: SYS 64738 will cold start and erase your program so SAVE it.

MAIN PROGRAM

This provides the working section of the completed program by loading the hi-res screen and machine code at the start. Type in correctly and SAVE but do not run unless the Data File has previously been entered and run.

Having entered both parts of the program, all that remains to be done is combine the two into one working program. Use the following procedure:

Ensure that the tape containing Main Program is fully rewound.
LOAD Data File into memory.
Verify tape to establish end of Main Program.
RUN Data File and when prompted to SAVE press Y; the m/c section will now be LOADed after Main Program to produce the completed copy.
Rewind and completed program should now LOAD by pressing SHIFT and RUN/STOP keys together.



VARIABLES USED

1. DATA FILE

A	*read DATA*
T	*total of DATA statements read*
SUM	*sum of DATA statements read*
A$	*keyboard entry*

2. MAIN PROGRAM

A	*counter to enter files from tape*
K	*number of keyboard entries*
T	*general counter*
CA	*number of correct answers*
CL	*number of figures in correct answer*
LL	*length of line indicates size of answer*
OP	*function selected*
QA	*number of questions answered*
QT	*answer to question set*
QX,QY	*random values to parts of question*
RP	*repeat counter*
SL	*range of random number*
SK	*skill level*
SU	*sum of questions by user*
SA	*screen address*
S1,S2	*character stores*
VR	*use in multiplication and division*
WR	*number of wrong answers*
$	*home*
R$	*cursor down, cursor left*
N$(x)	*string to print number between 0-9*
KEY$	*keyboard entry*

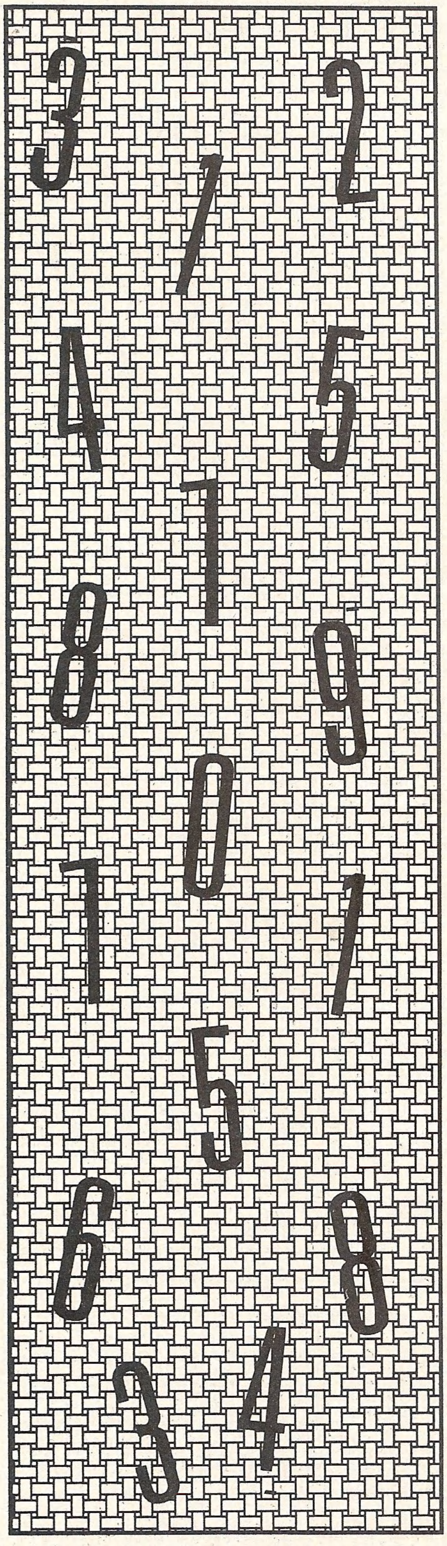
HOW IT RUNS

1. DATA FILE

14	screen and border black
16-28	text for DATA file
30-40	DATA entry for m/c routine to save completed program.
42-52	DATA for hi-res character set
54-64	DATA for sprite graphics
66-76	DATA for screen memory size
78-88	DATA entry for m/c routines
90-92	check for keyboard entry
94	clear screen
96	check for 'N'
98	m/c routine to save as a file between locations 32768-37887
100	m/c routine to save as a file between locations 38912-39361
102	m/c routine to display working page
104	check for keyboard entry
108	cold start

2. MAIN PROGRAM

14	clear screen to black
16	check for top of memory
18	initial delay
20	load DATA from tape as a file
22	load m/c from tape as a file
24	generate A$
26	generate R$
28	generate S$
30-50	generate number strings
52	call m/c routine to print screen
54-74	check keyboard entry for function required
78-86	store sprite character set for function set
88-94	check for skill level
102	set initial variables
104	calculate range of number and screen offset
106-116	calculate question and answer
118	check for 10 answered questions
120	calculate length of answer and line length
122	m/c routine to clear working area of screen and text for function set
128	set variables
130-134	print X
142-146	print Y
148	print line
156	clear keyboard buffer
158-172	check for keyboard entry
160	if DEL then delete one charactr
162	if RETURN then execute SUM
170	print keyboard entry
182	calculate SUM of entered answer
184	check for accuracy
186-200	execute correct procedure
208-226	error routine if less than 3 incorrect
228-248	error routine if 3 incorrect
256-264	recall author's name and print
266-290	print result of test
292-310	print text for end of test sequence
312-330	check for keyboard entry
332	check for f1
334	check for f3
336-340	end of test
342-350	set up for re-test



THE LITTLE PROFESSOR

The Little Professor is a registered trademark of Texas Instruments. Permission for publication of this mark has been obtained but there is no connection between Texas Instruments and the program printed here. No rights are implied for further use of the trademark in any program which may subsequently be published.

▷

Little Professor main program

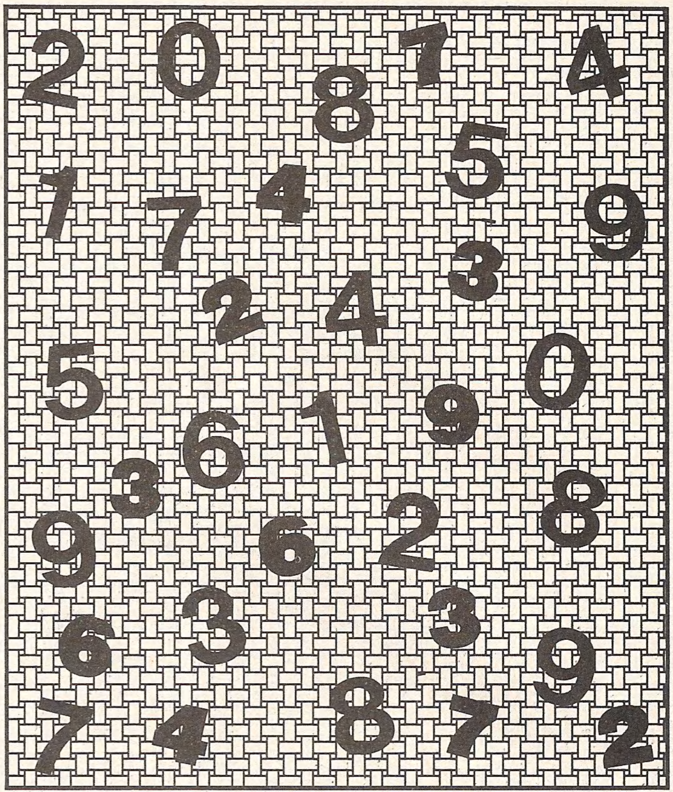
```
10 REM #### LITTLE PROFESSOR  MAIN PROGRAM ####
12 :
14 PRINTCHR$(147)CHR$(5)"LITTLE PROFESSOR":POKE53280,0:POKE53281,0
16 IFPEEK(52)<>160THEN24
18 IFA=0THENA=1:FORT=1TO2500:NEXT
20 IFA=1THENA=2:LOAD"PROFF DATA",1,1
22 IFA=2THENA=3:LOAD"PROFF M/C",1,1
24 FORT=1TO25:A$=A$+CHR$(17):NEXT:A$=CHR$(19)+A$
26 R$=CHR$(17)+CHR$(157)
28 S$=CHR$(32)+R$+CHR$(32)+R$+CHR$(32)+R$+CHR$(32)
30 N$(0)=CHR$(18)+CHR$(212)+R$+CHR$(213)+R$+CHR$(214)+R$+CHR$(215)
32 N$(1)=CHR$(18)+CHR$(216)+R$+CHR$(217)+R$+CHR$(218)+R$+CHR$(219)
34 N$(2)=CHR$(18)+CHR$(220)+R$+CHR$(221)+R$+CHR$(222)+R$+CHR$(223)
36 N$(3)=CHR$(18)+CHR$(224)+R$+CHR$(225)+R$+CHR$(226)+R$+CHR$(227)
38 N$(4)=CHR$(18)+CHR$(228)+R$+CHR$(229)+R$+CHR$(230)+R$+CHR$(231)
40 N$(5)=CHR$(18)+CHR$(232)+R$+CHR$(233)+R$+CHR$(234)+R$+CHR$(235)
42 N$(6)=CHR$(18)+CHR$(236)+R$+CHR$(237)+R$+CHR$(238)+R$+CHR$(239)
44 N$(7)=CHR$(18)+CHR$(240)+R$+CHR$(241)+R$+CHR$(242)+R$+CHR$(243)
46 N$(8)=CHR$(18)+CHR$(244)+R$+CHR$(245)+R$+CHR$(246)+R$+CHR$(247)
48 N$(9)=CHR$(18)+CHR$(248)+R$+CHR$(249)+R$+CHR$(250)+R$+CHR$(251)
50 SYS38912:RP=0
52 SA=1689:S1=6:S2=49
54 GETKEY$:IFKEY$<>""THENOP=ASC(KEY$)
56 IFRP=5THENSA=1689:S1=32:S2=32:S3=49
58 IFRP=10THENS1=6:S2=49
60 IFRP=15THENSA=1729:S1=32:S2=32:S3=51
62 IFRP=20THENS1=6:S2=51
64 IFRP=25THENSA=1769:S1=32:S2=32:S3=53
66 IFRP=30THENS1=6:S2=53
68 IFRP=35THENSA=1809:S1=32:S2=32:S3=55
70 IFRP=40THENS1=6:S2=55:RP=0
72 POKESA,S1:POKESA+1,S2
74 IFOP<133OROP>136THENRP=RP+1:GOTO54
76 POKESA,6:POKESA+1,S3:RP=0
78 POKE37891,44:POKE37892,6:POKE37893,224
80 IFOP=133THENPOKE37888,251:POKE37889,241:POKE37890,251
82 IFOP=134THENPOKE37888,242:POKE37889,243:POKE37890,244
84 IFOP=135THENPOKE37888,245:POKE37889,246:POKE37890,247
86 IFOP=136THENPOKE37888,248:POKE37889,249:POKE37890,250
88 GETKEY$:SL=VAL(KEY$)
90 IFRP=10THENPOKE1972,32:POKE1974,32:POKE1975,32:POKE1977,32
92 IFRP=20THENPOKE1972,49:POKE1974,20:POKE1975,15:POKE1977,57:RP=0
94 IFKEY$=""ORSL<1ORSL>9THENRP=RP+1:GOTO88
96 :
98 :REM #### RANDOM NUMBERS ####
100 :
102 CA=0:QA=0:VR=SL:WR=0:IFOP=133OROP=134THENVR=1
104 SK=10↑SQR(SL):OS=INT(SL/3)
106 QX=INT(RND(1)*SK)
108 QY=INT(RND(1)*SK/VR)+1
110 IFOP=133THENQT=QX+QY
112 IFOP=134THENQT=QX-QY:IFQT<0THEN106
114 IFOP=135THENQT=QX*QY
116 IFOP=136THENQT=INT(QX/QY):QX=QY*QT
118 QA=QA+1:IFQA>10THEN252
120 CL=LEN(STR$(QT)):LL=CL*2-3
122 SYS39232:SYS39312
124 REM #### PRINT X ####
126 :
128 K=0:L1=LL:XX=0:YY=0:XT=0
130 X=QX
132 X1=X-(INT(X/10)*10):X=(X-X1)/10:XX=XX+1
134 PRINTLEFT$(A$,10)SPC(33+OS-XX*2)CHR$(31)N$(X1):IFX<>0THEN132
136 :
138 REM #### PRINT Y ####
140 :
142 Y=QY
144 Y1=Y-(INT(Y/10)*10):Y=(Y-Y1)/10:YY=YY+1
146 PRINTLEFT$(A$,15)SPC(33+OS-YY*2)N$(Y1):IFY<>0THEN144
148 PRINTLEFT$(A$,19)SPC(32+OS-L1)CHR$(18)CHR$(210):L1=L1-1:IFL1<>0THEN148
150 :
152 REM #### KEYBOARD ENTRY ####
154 :
156 POKE198,0
158 GETKEY$
160 IFKEY$=CHR$(20)ANDK>0THENPRINTLEFT$(A$,20)SPC(33+OS-K*2)CHR$(146)S$:K=K-1
162 IFKEY$=CHR$(13)ANDK>0THEN180
164 IFKEY$=""ORKEY$<"0"ORKEY$>"9"THEN158
166 K=K+1:IFK<CL+1THENN=VAL(KEY$):N(K)=N
168 IFK=CL+1THENK=K-1:GOTO180
170 PRINTLEFT$(A$,20)SPC(33+OS-K*2)N$(N);
172 GOTO158
174 :
176 REM #### SUM ####
178 :
180 SUM=0:S=0
182 SUM=SUM+10↑S*N(S+1):S=S+1:IFS<KTHEN182
184 IFABS(QT-SUM)>.5THEN208
186 POKE37894,PEEK(37888):POKE37895,PEEK(37889)
188 POKE37896,PEEK(37890):POKE37897,PEEK(37891)
190 POKE37888,232:POKE37889,233:POKE37890,234:POKE37891,44:POKE37892,13
192 SYS39312
194 FORT=1TO2000:NEXT
196 POKE37888,PEEK(37894):POKE37889,PEEK(37895)
198 POKE37890,PEEK(37896):POKE37891,PEEK(37897)
200 POKE37892,6:CA=CA+1:WR=0:GOTO106
202 :
204 REM #### ERROR ####
206 :
208 POKE37894,PEEK(37888):POKE37895,PEEK(37889)
210 POKE37896,PEEK(37890):POKE37897,PEEK(37891)
212 POKE37888,235:POKE37889,236:POKE37890,237:POKE37891,54:POKE37892,2
214 SYS39312
216 FORT=1TO2000:NEXT
218 WR=WR+1:IFWR=3THEN228
220 POKE37888,PEEK(37894):POKE37889,PEEK(37895)
222 POKE37890,PEEK(37896):POKE37891,PEEK(37897)
224 POKE37892,6
226 GOTO122
228 POKE37888,238:POKE37889,239:POKE37890,240:POKE37891,54:POKE37892,9
230 SYS39312
232 XT=QT
234 X2=XT-(INT(XT/10)*10):XT=(XT-X2)/10:X3=X3+1
236 PRINTLEFT$(A$,20)SPC(33+OS-X3*2)N$(X2):IFXT<>0THEN234
238 PRINTLEFT$(A$,20)SPC(33+OS-(X3+1)*2)CHR$(146)S$
240 FORT=1TO2000:NEXT
242 POKE37888,PEEK(37894):POKE37889,PEEK(37895)
244 POKE37890,PEEK(37896):POKE37891,PEEK(37897)
246 POKE37892,6
248 WR=0:X3=0:GOTO106
250 :
252 REM #### FINISH ####
254 :
256 POKE37894,PEEK(37888):POKE37895,PEEK(37889)
258 POKE37896,PEEK(37890):POKE37897,PEEK(37891)
260 POKE37888,229:POKE37889,230:POKE37890,231:POKE37891,67:POKE37892,1
262 POKE37893,0
264 SYS39312:SYS39232
266 PRINTLEFT$(A$,9)SPC(25)CHR$(146)CHR$(31)"YOU HAVE NOW"
268 PRINTLEFT$(A$,11)SPC(25)"ANSWERED ALL"
270 PRINTLEFT$(A$,13)SPC(25)"10 QUESTIONS"
272 IFCA=10THENPRINTLEFT$(A$,15)SPC(30)N$(1)
274 IFCA=10THENPRINTLEFT$(A$,15)SPC(32)N$(0)
276 IFCA<>10THENPRINTLEFT$(A$,15)SPC(31)N$(CA)
278 PRINTLEFT$(A$,19)SPC(30)CHR$(18)CHR$(210)CHR$(210)CHR$(210)
280 PRINTLEFT$(A$,20)SPC(30)N$(1)
282 PRINTLEFT$(A$,20)SPC(32)N$(0)
284 IFCA=10THENPRINTLEFT$(A$,24)SPC(27)"EXCELLENT"
286 IFCA<10ANDCA>5THENPRINTLEFT$(A$,24)SPC(23)"VERY GOOD EFFORT"
288 IFCA<6THENPRINTLEFT$(A$,24)SPC(25)"NOT TOO GOOD"
290 FORT=1TO2000:NEXT
292 SYS39232:RP=0
294 PRINTLEFT$(A$,10)SPC(24)"YOU MAY CARRY"
296 PRINTLEFT$(A$,11)SPC(24)"ON  OR CHANGE"
298 PRINTLEFT$(A$,12)SPC(24)"FUNCTION  AND"
300 PRINTLEFT$(A$,13)SPC(24)"SKILL   LEVEL"
302 PRINTLEFT$(A$,14)SPC(24)"BY   PRESSING"
304 PRINTLEFT$(A$,15)SPC(24)"THE FOLLOWING"
306 PRINTLEFT$(A$,18)SPC(25)"F1      SAME"
308 PRINTLEFT$(A$,20)SPC(25)"F3   CHANGE"
310 PRINTLEFT$(A$,22)SPC(25)"F7      QUIT"
312 SA=1729:S1=6:S2=49
314 GETKEY$:IFKEY$<>""THENRT=ASC(KEY$)
316 IFRP=5THENSA=1729:S1=32:S2=32
318 IFRP=10THENS1=6:S2=49
320 IFRP=15THENSA=1809:S1=32:S2=32
322 IFRP=20THENS1=6:S2=51
324 IFRP=25THENSA=1889:S1=32:S2=32
326 IFRP=30THENS1=6:S2=55:RP=0
328 POKESA,S1:POKESA+1,S2
330 IFRT<>133ANDRT<>134ANDRT<>136THENRP=RP+1:GOTO314
332 IFRT=133THEN342
334 IFRT=134THENOP=0:RT=0:GOTO50
336 PRINTCHR$(147)CHR$(154)
338 POKE53248+21,0:POKE53248+24,21:POKE53280,14:POKE53281,6
340 END
342 POKE37888,PEEK(37894):POKE37889,PEEK(37895)
344 POKE37890,PEEK(37896):POKE37891,PEEK(37897)
346 POKE37892,6:POKE37893,224
348 SYS39232:SYS39312
350 RT=0:GOTO102
```

Little Professor DATA file

```
10 REM #### LITTLE PROFESSOR  DATA FILE ####
12 :
14 POKE53280,0:POKE53281,0
16 PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17)CHR$(5)SPC(12)"LITTLE PROFESSOR"
18 PRINTCHR$(17)CHR$(17)SPC(3)"THIS  BASIC  PROGRAM  IS  USED  TO"
20 PRINTCHR$(17)SPC(3)"GENERATE  THE  MACHINE  CODE  FILE"
22 PRINTCHR$(17)SPC(3)"FOR   HI-RESOLUTION   AND   SPRITE"
24 PRINTCHR$(17)SPC(3)"GRAPHICS, SCREEN  MEMORY  MAP  AND"
26 PRINTCHR$(17)SPC(3)"MACHINE CODE  ROUTINE  FOR  SCREEN"
28 PRINTCHR$(17)SPC(3)"DUMP   BY   THE   MAIN   PROGRAM ."
30 PRINTCHR$(17)CHR$(17)CHR$(17)SPC(3)"LOADING ......... M/C SAVE ROUTINE"
32 T=0:SUM=0
34 READA
36 IFABS(A)>255THENPRINT"ERROR IN DATA AT LINE NUMBER :"INT(T/16)*2+116:STOP
38 IFA<>-1THENPOKE32000+T,A:T=T+1:SUM=SUM+A:GOTO34
40 IFSUM<>8326ORT<>73THENPRINT"ERROR IN DATA STATEMENT !!":STOP
42 PRINTCHR$(145)SPC(3)"LOADING ..... HI-RES CHARACTER SET"
44 T=0:SUM=0
46 READA
48 IFABS(A)>255THENPRINT"ERROR IN DATA AT LINE NUMBER :"INT(T/16)*2+134:STOP
50 IFA<>-1THENPOKE32768+T,A:T=T+1:SUM=SUM+A:GOTO46
52 IFSUM<>199630ORT<>2048THENPRINT"ERROR IN DATA STATEMENT !!":STOP
54 PRINTCHR$(145)SPC(3)"LOADING ..... SPRITE CHARACTER SET"
56 T=0:SUM=0
58 READA
60 IFABS(A)>255THENPRINT"ERROR IN DATA AT LINE NUMBER :"INT(T/16)*2+398:STOP
62 IFA<>-1THENPOKE34816+T,A:T=T+1:SUM=SUM+A:GOTO58
64 IFSUM<>72768ORT<>1792THENPRINT"ERROR IN DATA STATEMENT !!":STOP
66 PRINTCHR$(145)SPC(3)"LOADING ........ SCREEN MEMORY MAP"
68 T=0:SUM=0
70 READA
72 IFABS(A)>255THENPRINT"ERROR IN DATA AT LINE NUMBER :"INT(T/16)*2+630:STOP
74 IFA<>-1THENPOKE36864+T,A:T=T+1:SUM=SUM+A:GOTO70
76 IFSUM<>89866ORT<>1000THENPRINT"ERROR IN DATA STATEMENT !!":STOP
78 PRINTCHR$(145)SPC(3)"LOADING ....... SCREEN M/C ROUTINE"
80 T=0:SUM=0
82 READA
84 IFABS(A)>255THENPRINT"ERROR IN DATA AT LINE NUMBER :"INT(T/16)*2+764:STOP
86 IFA<>-1THENPOKE38912+T,A:T=T+1:SUM=SUM+A:GOTO82
88 IFSUM<>54091ORT<>449THENPRINT"ERROR IN DATA STATEMENT !!":STOP
90 PRINTCHR$(145)SPC(3)" DO YOU WISH TO SAVE M/C [ Y/N ]  "
92 GETA$:IFA$<>"Y"ANDA$<>"N"THEN92
94 PRINTCHR$(147)
96 IFA$="N"THEN102
98 SYS32000,32768,37888,"PROFF DATA",01
100 SYS32000,38912,39362,"PROFF M/C",01
102 SYS38912
104 IFPEEK(197)=64THEN104
106 :
108 SYS64738
110 :
112 REM #### M/C ROUTINE TO SAVE DATA AS A FILE ON TAPE ####
114 :
116 DATA165,43,133,87,165,44,133,88,165,45,133,89,165,46,133,90
118 DATA32,253,174,32,138,173,32,247,183,165,20,133,43,165,21,133
120 DATA44,32,253,174,32,138,173,32,247,183,165,20,133,45,165,21
122 DATA133,46,32,253,174,32,86,225,165,87,133,43,165,88,133,44
124 DATA165,89,133,45,165,90,133,46,96
126 DATA-1
128 :
130 REM #### CHARACTER SET DATA ####
132 :
134 DATA192,255,255,255,255,255,255,255,126,102,102,254,230,230,230,0
136 DATA252,204,204,254,198,198,254,0,254,198,192,224,224,230,254,0
138 DATA254,198,198,230,230,230,254,0,126,96,96,248,224,224,254,0
140 DATA126,96,96,248,224,224,224,0,126,102,96,238,230,230,254,0
142 DATA102,102,102,254,230,230,230,0,24,24,24,56,56,56,56,0
144 DATA24,24,24,28,28,220,120,0,204,204,204,254,230,230,230,0
146 DATA96,96,96,224,224,224,254,0,127,73,73,201,201,201,201,0
148 DATA126,102,102,230,230,230,230,0,126,98,98,226,226,226,254,0
150 DATA126,102,102,254,224,224,224,0,126,98,98,226,234,254,12,0
152 DATA252,204,204,254,198,198,198,0,254,198,192,254,14,206,254,0
154 DATA254,48,48,56,56,56,56,0,102,102,102,230,230,230,254,0
156 DATA102,102,102,230,230,254,56,0,73,73,73,201,201,201,254,0
158 DATA198,198,198,56,198,198,198,0,198,198,198,254,56,56,56,0
160 DATA254,230,6,254,224,230,254,0,0,255,255,255,255,255,255,255
162 DATA3,255,255,255,255,255,255,255,0,1,1,1,3,2,2,6
164 DATA143,128,0,0,0,0,0,0,255,31,0,0,0,0,0,0
166 DATA0,0,0,0,0,0,0,0,255,255,63,0,0,0,0,0
168 DATA255,255,255,15,0,0,0,0,255,255,255,255,15,0,0,0
170 DATA255,255,255,255,255,15,0,0,255,255,255,255,255,255,15,0
172 DATA255,255,255,255,255,255,255,15,255,255,255,255,255,255,255,240
174 DATA255,255,255,255,255,255,240,0,255,255,255,255,255,240,0,0
```

```
176 DATA255,255,255,255,240,0,0,0,255,255,255,240,0,0,0,0
178 DATA255,255,240,0,0,0,0,0,255,240,0,0,0,0,0,0
180 DATA241,1,0,0,0,0,0,0,128,128,128,192,64,64,96
182 DATA126,102,102,230,230,230,254,0,24,24,24,56,56,56,56,0
184 DATA254,198,6,254,224,224,254,0,254,6,6,62,14,14,254,0
186 DATA198,198,198,254,14,14,14,0,254,198,192,252,28,28,252,0
188 DATA254,198,192,254,230,230,254,0,254,198,12,24,56,56,56,0
190 DATA124,100,100,254,230,230,254,0,254,198,198,254,14,206,254,0
192 DATA4,4,4,12,8,8,24,16,32,32,32,48,16,16,24,8
194 DATA0,0,0,0,1,3,6,28,16,16,48,240,32,48,60,126
196 DATA0,0,0,0,1,3,7,14,0,3,31,126,240,192,0,0
198 DATA0,192,248,126,15,3,0,0,0,0,0,128,192,224,112
200 DATA8,8,12,15,5,12,60,126,0,0,0,0,128,192,96,56
202 DATA0,0,0,0,1,1,3,2,48,33,103,199,143,31,63,63
204 DATA231,227,193,192,192,192,192,192,0,128,224,120,63,15,3,1
206 DATA255,255,192,192,192,192,192,192,255,255,0,0,0,0,0,1
208 DATA255,255,0,15,56,96,192,128,255,255,0,240,28,6,3,1
210 DATA255,255,0,0,0,0,0,128,252,248,48,48,48,48,48,48
212 DATA63,31,12,12,12,12,12,12,255,255,3,3,3,3,3,3
214 DATA0,1,7,30,124,240,192,128,231,195,131,3,3,3,3,3
216 DATA12,132,230,227,241,248,252,252,0,0,0,0,128,128,192,64
218 DATA6,4,12,8,9,9,25,19,127,127,255,255,255,255,255,255
220 DATA128,128,128,128,128,128,128,128,192,192,192,192,192,192,192,192
222 DATA1,3,2,2,2,2,3,1,3,15,31,63,63,127,127,127
224 DATA192,240,248,252,252,254,254,254,128,192,64,64,64,64,192,128
226 DATA48,48,48,48,48,48,48,48,12,12,12,12,12,12,12,12
228 DATA3,3,3,3,3,3,3,3,1,1,1,1,1,1,1,1
230 DATA254,254,255,255,255,255,255,255,96,32,48,16,144,144,152,200
232 DATA19,19,19,19,19,19,19,19,192,192,192,192,192,192,255,255
234 DATA1,0,0,0,0,0,255,255,255,255,127,63,31,7,255,255
236 DATA255,255,254,252,248,224,255,255,128,0,0,0,0,0,255,255
238 DATA48,48,48,48,48,48,240,240,12,12,12,12,12,12,15,15
240 DATA3,3,3,3,3,3,255,255,1,1,1,1,1,1,1,1
242 DATA200,200,200,200,200,200,200,200,25,9,8,12,4,4,6,2
244 DATA255,255,255,255,255,127,127,63,0,128,128,128,192,64,64,96
246 DATA0,1,1,1,3,2,2,6,255,255,255,255,255,254,254,252
248 DATA152,144,16,48,32,32,96,64,3,1,1,0,0,0,0,0
250 DATA63,31,31,143,199,99,49,24,128,128,128,128,128,192,192,64
252 DATA0,0,7,15,15,7,7,3,0,0,0,128,224,240,248,254
254 DATA32,48,16,24,14,3,0,0,32,48,24,28,119,193,0,0
256 DATA0,0,0,0,0,128,248,15,0,0,0,0,0,1,31,240
258 DATA4,12,24,56,238,131,0,0,4,12,8,24,112,192,0,0
260 DATA0,0,0,1,7,15,31,127,0,0,224,240,240,224,224,192
262 DATA1,1,1,1,1,3,7,6,252,248,249,241,225,198,140,24
264 DATA192,128,128,0,0,0,0,0,12,6,3,1,0,0,0,0
266 DATA64,64,64,224,32,32,32,48,3,1,0,0,0,0,0,0
268 DATA255,255,255,255,127,63,31,15,128,224,248,254,255,255,255,255
270 DATA0,0,0,0,192,252,255,255,0,0,0,0,0,0,192,255
272 DATA0,0,0,0,0,0,3,255,0,0,0,0,3,63,255,255
274 DATA1,7,31,127,255,255,255,255,255,255,255,255,254,252,248,240
276 DATA192,128,0,0,0,0,0,0,2,2,2,7,4,4,4,12
278 DATA48,96,192,128,0,0,0,0,16,16,16,16,16,24,8,8
280 DATA15,3,9,8,24,16,16,16,255,255,255,255,127,63,15,7
282 DATA255,255,255,255,254,252,248,224,240,192,144,16,24,8,8,8
284 DATA8,8,8,8,8,24,16,16,8,8,12,4,4,6,2,2
286 DATA16,16,24,8,8,8,8,8,3,0,0,0,0,0,0,0
288 DATA255,255,63,31,3,0,2,3,255,255,255,255,255,255,31,1
290 DATA255,255,255,255,255,255,252,128,255,255,252,248,192,0,64,192
292 DATA192,0,0,0,0,0,0,0,8,8,24,16,16,16,16,16
294 DATA255,255,255,255,255,255,255,255,16,16,48,32,32,96,64,64
296 DATA3,1,1,1,1,0,0,0,0,0,0,128,128,128,128
298 DATA8,8,12,4,6,2,3,1,1,1,0,0,0,0,0,0
300 DATA0,128,128,192,96,48,24,14,0,1,1,3,6,12,24,112
302 DATA128,128,0,0,0,0,0,0,16,16,48,32,96,64,192,128
304 DATA0,0,0,0,1,1,1,1,192,128,128,128,128,0,0,0
306 DATA224,48,24,14,3,1,0,0,0,0,0,0,0,128,192,112
308 DATA1,1,0,0,0,0,0,0,0,128,192,64,96,48,24,12
310 DATA3,0,0,0,0,0,0,0,192,120,15,0,0,0,0,0
312 DATA3,30,240,0,0,0,0,0,192,0,0,0,0,0,0,0
314 DATA0,1,3,2,6,12,24,48,128,128,0,0,0,0,0,0
316 DATA0,0,0,0,0,1,3,6,7,12,24,48,192,128,0,0
318 DATA28,6,3,0,0,0,0,0,0,0,128,224,56,14,3,0
320 DATA0,0,0,0,0,0,128,224,6,3,1,0,0,0,0,0
322 DATA0,0,128,192,96,56,14,3,0,0,0,0,0,0,0,128
324 DATA0,0,0,0,0,0,0,1,0,0,1,3,6,28,112,192
326 DATA96,192,128,0,0,0,0,0,0,0,0,0,0,0,1,7
328 DATA0,0,1,7,28,112,192,0,56,96,192,0,0,0,0,0
330 DATA60,7,1,0,0,0,0,0,0,0,224,62,3,0,0,0
332 DATA0,0,0,0,192,126,3,0,224,56,15,0,0,0,252,7
334 DATA0,0,128,254,3,0,0,255,0,0,1,63,224,0,0,255
336 DATA7,28,240,0,0,0,63,224,0,0,0,0,3,126,192,0
338 DATA0,0,7,124,192,0,0,0,126,192,128,0,0,0,0,0
340 DATA255,128,128,128,128,128,128,128,128,128,128,128,128,128,128,255
342 DATA255,1,1,1,1,1,1,1,1,1,1,1,1,1,1,255
344 DATA255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255
346 DATA126,126,126,126,102,102,102,102,102,102,102,102,230,230,230,230
348 DATA230,230,230,230,230,230,230,230,254,254,254,254,0,0,0,0
350 DATA24,24,24,24,24,24,24,24,24,24,24,24,56,56,56,56
352 DATA56,56,56,56,56,56,56,56,56,56,56,56,0,0,0,0
354 DATA254,254,254,254,198,198,198,198,6,6,6,6,254,254,254,254
356 DATA224,224,224,224,224,224,224,224,254,254,254,254,0,0,0,0
358 DATA254,254,254,254,6,6,6,6,6,6,6,6,62,62,62,62
360 DATA14,14,14,14,14,14,14,14,254,254,254,254,0,0,0,0
362 DATA198,198,198,198,198,198,198,198,198,198,198,198,254,254,254,254
364 DATA14,14,14,14,14,14,14,14,14,14,14,14,0,0,0,0
366 DATA254,254,254,254,230,230,230,230,192,192,192,192,252,252,252,252
368 DATA28,28,28,28,28,28,28,28,252,252,252,252,0,0,0,0
370 DATA254,254,254,254,198,198,198,198,192,192,192,192,254,254,254,254
372 DATA230,230,230,230,230,230,230,230,254,254,254,254,0,0,0,0
374 DATA254,254,254,254,198,198,198,198,12,12,12,12,24,24,24,24
376 DATA56,56,56,56,56,56,56,56,56,56,56,56,0,0,0,0
378 DATA126,126,126,126,102,102,102,102,102,102,102,102,254,254,254,254
380 DATA230,230,230,230,230,230,230,230,254,254,254,254,0,0,0,0
382 DATA254,254,254,254,198,198,198,198,198,198,198,198,254,254,254,254
384 DATA14,14,14,14,206,206,206,206,254,254,254,254,0,0,0,0
386 DATA0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,255
388 DATA255,255,255,255,254,252,248,240,255,255,255,255,127,63,31,15
390 DATA-1
392 :
394 REM #### SPRITE DATA ####
396 :
398 DATA96,0,0,96,0,0,96,0,0,96,24,24,96,24,24,96
400 DATA24,24,96,0,126,96,0,126,96,0,126,96,56,24,96,56
402 DATA24,96,56,24,96,24,24,96,24,24,96,24,24,96,24,24
404 DATA96,24,24,96,24,24,126,60,14,126,60,14,126,60,14,0
406 DATA0,0,0,0,0,0,0,0,24,56,0,24,56,0,24
408 DATA56,0,126,24,60,126,24,60,126,24,60,24,24,102,24,24
410 DATA102,24,24,102,24,24,126,24,24,126,24,24,126,24,24,96
412 DATA24,24,96,24,24,96,14,60,60,14,60,60,14,60,60,0
414 DATA124,0,0,124,0,0,124,0,0,102,0,0,102,0,0,102
416 DATA0,0,102,124,60,102,124,60,102,124,60,124,102,102,124,102
418 DATA102,124,102,102,96,96,102,96,96,102,96,96,102,96,96,102
420 DATA96,96,102,96,96,102,96,96,60,96,96,60,96,96,60,0
422 DATA0,0,0,0,0,0,0,0,14,0,0,14,0,0,14
424 DATA0,0,24,60,62,24,60,62,24,60,62,62,102,96,62,102
426 DATA96,62,102,96,24,126,60,24,126,60,24,126,60,24,96,6
428 DATA24,96,6,24,96,6,24,60,60,24,60,60,24,60,60,0
430 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
432 DATA0,0,62,60,124,62,60,124,62,60,124,96,102,102,96,102
434 DATA102,96,102,102,60,102,96,60,102,96,60,102,96,6,102,96
436 DATA6,102,96,6,102,96,124,60,96,124,60,96,124,60,96,0
438 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
440 DATA0,24,0,0,24,96,0,24,96,0,24,96,0,24,96,0
442 DATA24,121,176,24,121,176,24,109,176,24,109,176,24,108,240,24
444 DATA108,240,24,120,96,31,120,96,31,1,192,0,1,192,0,0
446 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
448 DATA0,14,0,0,14,0,0,27,0,0,27,0,0,27,56,240
450 DATA27,109,128,31,109,128,27,124,224,27,124,224,27,96,48,27
452 DATA96,48,27,57,224,27,57,224,27,0,0,0,0,0,0
454 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
456 DATA0,0,0,0,0,113,192,0,113,192,0,48,199,158,48,199
458 DATA158,48,192,219,48,192,219,48,199,219,48,199,219,48,204,219
460 DATA48,204,219,121,231,219,121,231,219,0,0,0,0,0,0
462 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
464 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
466 DATA0,0,0,0,0,0,0,0,0,99,126,96,99,96,96
468 DATA99,96,96,107,120,96,127,96,96,119,96,96,99,126,126,0
470 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
472 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
474 DATA0,0,0,0,0,0,0,0,0,96,0,120,96,0,108
476 DATA96,0,102,96,0,102,96,0,102,96,0,108,126,0,120,0
478 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
480 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
482 DATA0,0,0,0,0,0,0,0,0,60,102,126,102,118,96
484 DATA102,126,96,102,126,120,102,110,96,102,102,96,60,102,126,0
486 DATA0,0,0,0,0,0,0,0,0,0,99,0,0,99,0
488 DATA0,99,0,0,107,0,0,127,0,0,119,0,0,99,0,0
490 DATA0,0,0,0,0,0,0,0,0,126,124,102,24,102,102
492 DATA24,102,102,24,124,60,24,120,24,24,108,24,24,102,24,0
494 DATA0,0,0,0,0,0,0,0,124,60,102,102,102,118,102
496 DATA102,126,124,102,126,120,102,110,108,102,102,102,60,102,0,0
498 DATA0,0,0,0,0,0,0,0,0,0,24,60,0,60,102
500 DATA0,102,96,0,126,110,0,102,102,0,102,102,0,102,60,0
502 DATA0,0,0,0,0,0,0,0,60,0,0,102,0,0,96
504 DATA0,0,110,0,0,102,0,0,102,0,0,60,0,0,0,0
506 DATA0,0,0,0,0,0,0,0,0,24,60,102,60,24,118
508 DATA102,24,126,126,24,126,102,24,110,102,24,102,102,60,102,0
510 DATA0,0,0,0,0,0,0,0,0,60,60,0,102,102,0
512 DATA96,102,0,96,102,0,96,102,0,96,102,0,60,60,0,0
514 DATA0,0,0,0,0,0,0,0,0,24,102,60,60,118,102
516 DATA102,126,96,126,126,60,102,110,6,102,102,102,102,102,60,0
518 DATA0,0,0,0,0,0,0,0,124,124,126,102,102,96,102
520 DATA102,96,124,124,120,120,120,96,108,108,96,102,102,126,0,0
522 DATA0,0,0,0,0,0,0,0,0,99,126,124,99,96,102
524 DATA99,96,102,107,120,124,127,96,120,119,96,108,99,126,102,0
526 DATA0,0,0,0,0,0,0,0,60,126,0,102,24,0,96
528 DATA24,0,96,24,0,96,24,0,102,24,0,60,24,0,0,0
530 DATA0,0,0,0,0,0,0,0,0,0,60,60,0,24,102
532 DATA0,24,96,0,24,60,0,24,6,0,24,102,0,60,60,0
534 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
536 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
538 DATA0,0,0,0,0,0,0,0,0,24,120,120,60,108,108
540 DATA102,102,102,126,102,102,102,102,102,102,108,108,102,120,120,0
542 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
544 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
546 DATA0,0,0,0,0,0,0,0,0,3,198,103,6,102,102
548 DATA6,6,102,3,198,103,0,102,102,6,102,102,3,195,199,0
550 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
552 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
554 DATA0,0,0,0,0,0,0,0,0,199,231,193,97,134,99
556 DATA97,134,102,129,135,199,97,135,134,97,134,198,193,134,102,0
558 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
560 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
562 DATA0,0,0,0,0,0,0,0,0,131,199,224,198,97,128
564 DATA102,1,128,230,1,128,102,1,128,102,97,128,99,193,128,0
566 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
568 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
570 DATA0,0,0,0,0,0,0,0,0,6,54,102,7,118,102
572 DATA7,246,102,6,182,102,6,54,102,6,54,102,6,51,199,0
574 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
576 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
578 DATA0,0,0,0,0,0,0,0,0,7,227,199,1,129,134
580 DATA1,129,134,1,129,135,1,129,134,1,129,134,225,131,198,0
582 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
584 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
586 DATA0,0,0,0,0,0,0,0,0,198,6,96,102,6,96
588 DATA102,6,96,198,3,192,6,1,128,6,1,128,7,225,128,0
590 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
592 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
594 DATA0,0,0,0,0,0,0,0,0,0,7,131,0,6,193
596 DATA0,6,97,0,6,97,0,6,97,0,6,193,0,7,131,0
598 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
600 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
602 DATA0,0,0,0,0,0,0,0,0,198,99,199,134,97,134
604 DATA134,97,134,134,97,134,134,97,134,131,193,134,193,131,199,0
606 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
608 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
610 DATA0,0,0,0,0,0,0,0,0,135,224,0,198,0,0
612 DATA102,0,0,103,128,0,102,0,0,198,0,0,135,224,0,0
614 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
616 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
618 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
620 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
622 DATA-1
624 :
626 REM #### SCREEN MEMORY MAP ####
628 :
630 DATA206,210,210,210,210,210,210,210,210,210,210,210,210,210,210,210
632 DATA210,210,210,210,210,210,210,210,210,210,210,210,210,210,210,210
634 DATA210,210,210,210,210,210,210,208,86,32,32,160,160,160,160,160
636 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,160,160
638 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,32,32,107
640 DATA86,32,32,160,160,160,160,160,160,160,160,160,160,160,160,160
642 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,160,160
644 DATA160,160,160,160,160,32,32,107,86,32,32,160,160,160,160,160
646 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,160,160
648 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,32,32,107
650 DATA86,32,32,160,160,160,160,160,160,160,160,160,160,160,160,160
652 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,160,160
654 DATA160,160,160,160,160,32,32,107,86,32,32,160,160,160,160,160
656 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,160,160
658 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,32,32,107
660 DATA86,32,32,32,32,32,32,32,32,32,32,32,32,32,32
662 DATA32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
664 DATA32,32,32,32,32,32,32,107,86,32,160,160,160,160,160,160
666 DATA160,160,160,160,160,160,160,160,160,160,160,160,160,160,32,8
668 DATA5,18,5,32,9,19,32,1,32,19,9,13,16,12,5,107
670 DATA86,32,32,32,0,27,27,27,27,27,27,27,27,27,27,27
672 DATA27,27,27,28,32,32,32,20,5,1,3,8,9,14,7,32
674 DATA1,9,4,32,6,15,18,107,86,32,32,29,30,31,33,34
676 DATA35,36,37,38,39,40,41,42,43,44,45,46,47,32,32,25
678 DATA15,21,14,7,32,32,32,3,8,9,12,4,18,5,14,107
680 DATA86,32,32,58,32,32,32,32,32,32,32,32,32,32,32,32
682 DATA32,32,32,32,59,32,32,32,32,32,32,32,32,32,32,32
684 DATA32,32,32,32,32,32,32,107,86,32,60,61,32,32,32,32
686 DATA32,32,62,63,64,65,32,32,32,32,32,32,66,67,32,25
688 DATA15,21,32,1,18,5,32,19,5,20,32,32,20,5,14,107
690 DATA86,68,69,70,71,72,73,74,75,76,77,32,32,78,73,74
692 DATA75,76,79,80,81,82,83,17,21,5,19,20,9,15,14,19
694 DATA32,32,32,32,32,2,25,107,86,84,85,86,32,87,88,89
696 DATA90,91,92,32,32,93,88,89,90,91,94,32,95,96,97,19
698 DATA5,12,5,3,20,9,15,14,32,15,6,32,20,8,5,107
700 DATA86,98,160,86,32,99,100,101,102,103,104,32,32,105,100,101
702 DATA102,103,106,32,107,160,108,6,15,12,12,15,23,9,14,7
704 DATA32,32,32,11,5,25,19,107,86,109,110,86,32,32,32,32
706 DATA32,32,111,32,32,112,32,32,32,32,32,32,107,113,114,32
708 DATA32,32,32,32,32,32,32,32,32,32,32,32,32,32,107
710 DATA86,115,116,117,32,32,118,119,32,120,121,122,123,124,125,32
712 DATA126,127,32,32,128,129,130,32,32,6,49,32,32,32,32,32
714 DATA32,32,1,4,4,32,32,107,86,32,131,132,32,32,133,134
```

```
716 DATA135,136,137,32,32,138,139,140,141,142,32,32,143,144,32,32
718 DATA32,6,51,32,32,19,21,2,20,18,1,3,20,32,32,107
720 DATA86,32,32,145,32,32,32,146,147,160,160,160,160,160,160,148
722 DATA149,32,32,32,150,32,32,32,32,6,53,32,32,13,21,12
724 DATA20,9,16,12,25,32,32,107,86,32,32,151,32,32,32,152
726 DATA153,154,155,254,255,156,157,158,159,32,32,32,161,32,32,32
728 DATA32,6,55,32,32,32,32,4,9,21,9,4,5,32,32,107
730 DATA86,32,32,162,163,32,32,164,32,165,166,32,32,167,168,32
732 DATA169,32,32,170,171,32,32,32,32,32,32,32,32,32,32,32
734 DATA32,32,32,32,32,32,32,107,86,32,32,32,172,173,32,174
736 DATA175,32,176,177,178,179,32,180,181,32,182,183,32,32,32,32
738 DATA32,19,11,9,12,12,32,32,12,5,22,5,12,32,32,107
740 DATA86,32,32,32,32,184,185,186,187,188,189,32,32,190,191,192
742 DATA193,194,195,32,32,32,32,32,32,32,32,211,211,211,211,211
744 DATA211,211,211,32,32,32,32,107,86,32,32,32,32,32,32,196
746 DATA197,198,199,200,201,202,203,204,205,32,32,32,32,32,32,32
748 DATA32,32,32,86,49,32,20,15,32,57,107,32,32,32,32,107
750 DATA207,211,211,211,211,211,211,211,211,211,211,211,211,211,211,211
752 DATA211,211,211,211,211,211,211,211,211,211,211,253,253,253,253,253
754 DATA253,253,253,211,211,211,211,209
756 DATA-1
758 :
760 REM #### M/C ROUTINE FOR DATA TRANSFER ####
762 :
764 DATA169,40,133,52,133,56,162,0,189,0,128,157,0,48,189,0
766 DATA129,157,0,49,189,0,130,157,0,50,189,0,131,157,0,51
768 DATA189,0,132,157,0,52,189,0,133,157,0,53,189,0,134,157
770 DATA0,54,189,0,135,157,0,55,189,0,136,157,0,56,189,0
772 DATA137,157,0,57,189,0,138,157,0,58,189,0,139,157,0,59
774 DATA189,0,140,157,0,60,189,0,141,157,0,61,189,0,142,157
776 DATA0,62,232,240,3,76,8,152,169,240,45,24,208,9,12,141
778 DATA24,208,162,0,189,0,144,157,0,4,189,250,144,157,250,4
780 DATA189,244,145,157,244,5,189,238,146,157,238,6,169,6,157,0
782 DATA216,157,250,216,157,244,217,157,238,218,232,224,250,240,3,76
784 DATA116,152,162,0,169,14,157,43,216,157,83,216,157,123,216,157
786 DATA163,216,157,203,216,157,243,216,232,224,34,240,3,76,166,152
788 DATA169,14,141,32,208,169,15,141,33,208,169,67,141,1,208,141
790 DATA3,208,141,5,208,141,7,208,141,9,208,141,11,208,141,13
792 DATA208,141,15,208,169,60,141,0,208,169,84,141,2,208,169,132
794 DATA141,4,208,169,156,141,6,208,169,180,141,8,208,169,236,141
796 DATA10,208,169,4,141,12,208,169,28,141,14,208,169,1,162,0
798 DATA157,39,208,232,224,8,240,3,76,16,153,169,255,141,21,208
800 DATA169,192,141,16,208,162,0,160,224,152,157,248,7,200,232,224
802 DATA8,240,3,76,41,153,169,0,141,23,208,141,29,208,96,234
804 DATA162,0,169,32,157,47,5,157,87,5,157,127,5,157,167,5
806 DATA157,207,5,157,247,5,157,31,6,157,71,6,157,111,6,157
808 DATA151,6,157,191,6,157,231,6,157,15,7,157,55,7,157,95
810 DATA7,157,135,7,157,175,7,169,211,157,215,7,232,224,16,240
812 DATA3,76,66,153,96,234,234,234,234,234,234,234,234,234,234,234
814 DATA173,0,148,141,253,7,173,1,148,141,254,7,173,2,148,141
816 DATA255,7,173,3,148,141,11,208,141,13,208,141,15,208,173,4
818 DATA148,141,44,208,141,45,208,141,46,208,173,5,148,141,23,208
820 DATA96,-1
```



SUBSCRIPTIONS

Whatever your interest in the Commodore range of computers and peripherals, you cannot afford to miss a single issue of YOUR COMMODORE.

To avoid disappointment, fill out the form below and send it off to the Subscriptions department. You know it makes sense!

SUBSCRIPTION ORDER FORM

Cut out and SEND TO:
YOUR COMMODORE
INFONET LTD., TIMES HSE,
179 THE MARLOWES,
HEMEL HEMPSTEAD,
HERTS, HP1 1BB

Please commence my personal subscription YOUR COMMODORE with the ....issue

SUBSCRIPTION RATES (tick as appropriate)	**£14.70 for 12 issues UK**	☐
	£17.20 for 12 issues Overseas surface	☐
	£50.20 for 12 issues Overseas Air Mail	☐

I am enclosing my (delete as necessary)
Cheque/Postal Order/International Money Order for£ ____________
(made payable to ASP Ltd)
OR
Debit my Access/Barclaycard★
(★delete as necessary)

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Please use BLOCK CAPITALS and include post codes.

Name (Mr/Mrs/Miss)...........................
(delete accordingly)

Address ....

....

Signature ________________ *Date*..........

ALLEN WEBB

Great Graphics

Memory-efficient routines that will stand on their own or together.

Although the graphic capabilities of the 64 are powerful and can be extended further with graphic packages, it is unfortunate that most such packages steal a large hunk of RAM.

If you only want to use one or two bits of the package this can be impracticable. So here I offer you routines which can be used on their own with only a minimal overhead in RAM.

There seems to be a lot of inverted snobbery concerning hi-res graphics these days. Whilst their use is ideal for graph plotting routines, they tend to be RAM hungry, often slow and with limitations. For instance, can you create an animated drawing of a river using hi-res graphics? It's easy using redefined characters!

So my routines will offer you easy alternatives to hi-res and, whilst they all operate independently, they can be used together.

MEDIUM RESOLUTION

This routine gives pseudo teletext capabilities by enabling you to plot points and draw lines to an 80 by 50 resolution. Two simple commands are available:

To clear or set a point: SYS 49152,X,Y,Colour,Type.

To draw a line: SYS 49155,X1,Y1,X2,Y2,Colour,Type. In both cases Type=1 draws and Type=0 erases.

These routines assume that the screen memory is in its normal place and they will not work if you move the screen. For fast drawing it is best to use them directly from machine code; details later. Listing 1 gives a BASIC loader for these routines; Demo 1 shows how they work.

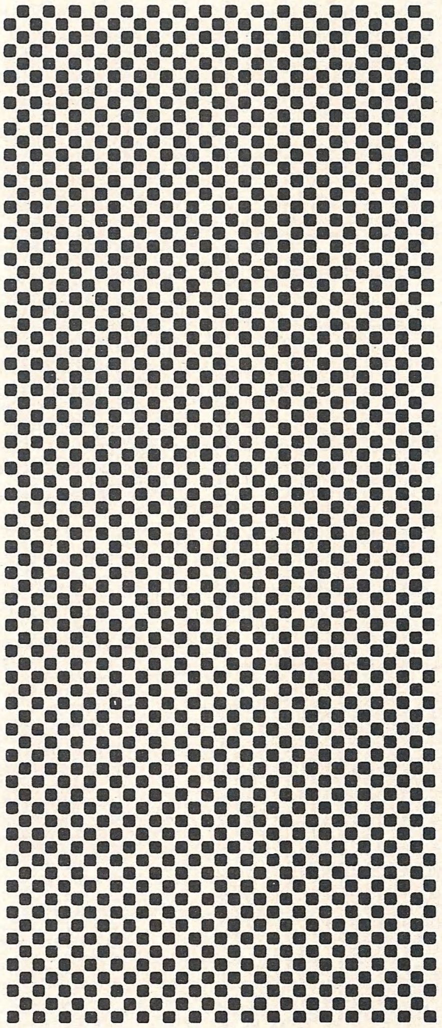
USER DEFINED CHARACTERS

Most computers have the option for user defined characters and the 64 is no exception, except that there is no command to create them.

The first problem in creating such a routine is knowing where to put the new characters. To maintain compatibility with the plotting routines we are compelled to keep the screen in its normal place. To minimise loss of RAM the new characters are put in a block from 2048 to 4095, with the start of BASIC shunted up to 4096.

There are two ways of doing this. First, type in as a direct command: POKE 44,16: POKE 256*16,0: NEW. Or you can use one of my routines to it for you.

Both of these methods will destroy any BASIC program in memory, so it is wise to do this first. Protect the graphics by calling SYS 49926,X — any value apart from 255 for X will raise BASIC but 255 will auto load the next program on tape, thus giving no means of booting a sequence of programs.



Then switch in the new graphics set and copy the specified characters from RAM with SYS 49920, Set where Set=0 gives upper case and 1 gives lower. A message now comes up to tell you this command has finished but by POKEing location 1000 with 255 you can supress this.

To define the shape of a character N, call SYS 49923 [N], B1,B2,B3,B4,B5,B6,B7,B8 — the B parameters specify the shape. Then just enable your user defined set with SYS 49929 and disable with SYS 49932. Listing 2 gives you a BASIC loader for these routines; Demo 2 some examples.

BILLBOARD DISPLAYS

This next routine is perhaps of more limited value but it can be linked with the plotting routines to produce billboard type displays. It draws a large character at a specified position on screen. These macro characters are built up from the standard CBM set and are 4 x 4 characters large.

Any number of letter can be called with SYS 50432,X, Y,CH,COLOUR, where X is the column where the character starts (0-39), Y the row where the top left hand corner starts (0-24), CH is the ASCII value of the character and COLOUR is the colour required. Listing 3 gives the BASIC loader; Demo 3 the use.

SCROLLS AND ROLLS

The final routines give a full range of scroll and roll facilities. Scrolling moves the screen one character in the required direction and puts a blank line at the other end of the screen; rolling down the same but shifts the moved line to the other edge of the screen. Both shift the screen and colour memories so that the screen appearance is maintained.

Call the routines by: SYS 52742 for scroll down; SYS 51826 for roll down; SYS 52123 for scroll left; SYS 52182 for roll left; SYS 51992 for scroll right; and SYS 52049 for roll right. Listings 4 and 5 load these routines; Demo 4 shows what is possible with them.

MACHINE CODE

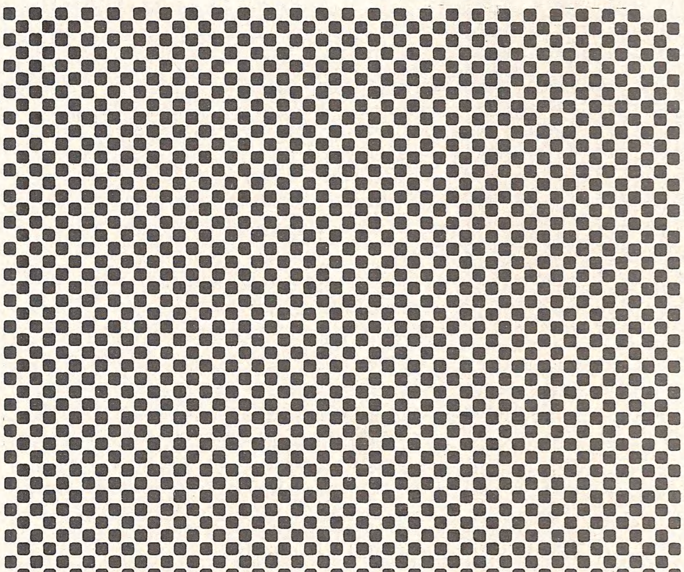
For machine code addicts, here are some details of the machine code aspects of low-res plotting. The interfacing of the plotting routines to your own machine code routines is a simple matter with these details:

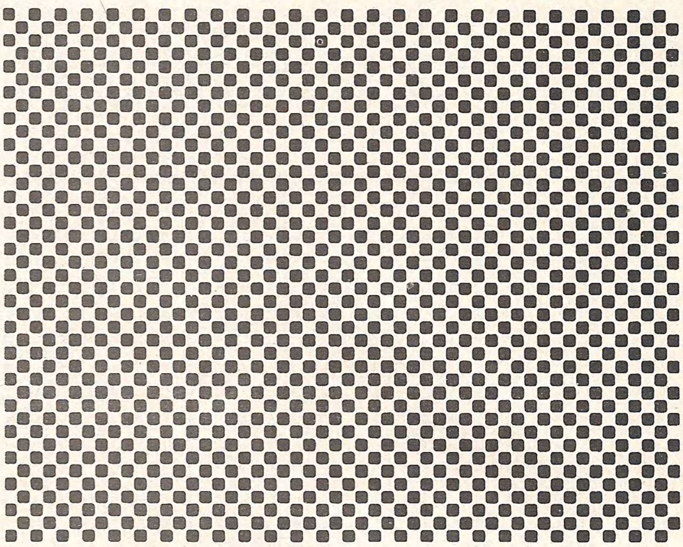
Point plot: X in location 900; Y in 901; Colour in 786; Point type in 906; Entry point $C006.

Line plot: X1 in location 902; Y1 in 903; X2 in 904; Y2 in 905; Colour in 786; Line type in 907; Entry point $COD1.

```
10 LISTING 1
20 REM
30 REM
40 DATA76,51,194,76,111,194,72,152,72,138,72,169,0,141,88,3,141,64,3,133,252
50 DATA173,133,3,133,251,165,251,201,50,144,3,238,88,3,173,132,3,201,80,144
60 DATA3,238,88,3,173,88,3,240,3,76,187,192,169,49,56,229,251,133,251,78,132
70 DATA3,46,64,3,70,251,46,64,3,6,251,6,251,6,251,165,251,6,251,38,252,6,251
80 DATA38,252,24,101,251,133,251,165,252,105,4,133,252,174,64,3,169,1,141,64
90 DATA3,224,0,240,6,14,64,3,202,144,246,172,132,3,177,251,162,0,221,193,192
100 DATA240,13,232,224,16,144,246,169,4,141,88,3,76,187,192,173,138,3,240,8
110 DATA138,13,64,3,24,170,144,13,173,64,3,73,255,141,64,3,138,45,64,3,170,189
120 DATA193,192,172,132,3,145,251,24,165,252,105,212,133,252,173,18,3,145,251
130 DATA104,170,104,168,104,96,32,126,123,97,124,226,255,236,108,127,98,252
140 DATA225,251,254,160,72,152,72,138,72,169,255,141,138,3,173,134,3,201,80
150 DATA144,5,160,1,76,42,194,173,136,3,201,80,144,5,160,2,76,42,194,173,135
160 DATA3,201,50,144,5,160,3,76,42,194,173,137,3,201,50,144,5,160,4,76,42,194
170 DATA173,136,3,56,237,134,3,141,65,3,173,137,3,56,237,135,3,141,66,3,169
180 DATA1,141,67,3,141,68,3,173,66,3,16,10,160,255,140,67,3,73,255,24,105,1
190 DATA141,69,3,173,65,3,16,10,160,255,140,68,3,73,255,24,105,1,141,70,3,173
200 DATA70,3,56,237,69,3,141,71,3,16,35,169,255,141,72,3,169,0,141,73,3,173
210 DATA69,3,141,74,3,173,70,3,141,75,3,173,66,3,48,5,169,1,141,72,3,76,154
220 DATA193,169,0,141,72,3,169,255,141,73,3,173,70,3,141,74,3,173,69,3,141
230 DATA75,3,173,65,3,48,5,169,1,141,73,3,173,74,3,141,76,3,56,237,75,3,141
240 DATA78,3,173,75,3,141,77,3,78,74,3,173,75,3,56,237,74,3,141,79,3,173,134
250 DATA3,141,132,3,173,135,3,141,133,3,173,139,3,208,4,169,0,240,13,201,1
260 DATA208,4,169,1,208,5,173,138,3,73,255,141,138,3,32,6,192,173,79,3,16,30
270 DATA24,109,77,3,141,79,3,173,134,3,24,109,73,3,141,134,3,173,135,3,24,109
280 DATA72,3,141,135,3,76,32,194,56,237,78,3,141,79,3,173,134,3,24,109,68,3
290 DATA141,134,3,173,135,3,24,109,67,3,141,135,3,206,76,3,48,3,76,186,193
300 DATA160,0,140,89,3,104,170,104,168,104,96,32,253,174,32,138,173,32,247
310 DATA183,165,20,141,132,3,32,253,174,32,138,173,32,247,183,165,20,141,133
320 DATA3,32,253,174,32,138,173,32,247,183,165,20,141,18,3,32,253,174,32,138
330 DATA173,32,247,183,165,20,141,138,3,76,6,192,96,32,253,174,32,138,173,32
340 DATA247,183,165,20,141,134,3,32,253,174,32,138,173,32,247,183,165,20,141
350 DATA135,3,32,253,174,32,138,173,32,247,183,165,20,141,136,3,32,253,174
360 DATA32,138,173,32,247,183,165,20,141,137,3,32,253,174,32,138,173,32,247
370 DATA183,165,20,141,18,3,32,253,174,32,138,173,32,247,183,165,20,141,139
380 DATA3,76,209,192
390 T=0:FORI=49152 TO 49861
400 READ X: T=T+X: POKEI,X:NEXT
410 IF T<>75395 THEN PRINT"DATA ERROR"
420 REM********************************
430 REM*   MEDIUM RESOLUTION PLOT     *
440 REM* THIS OCCUPIES $C000-$C2C5    *
450 REM*                              *
460 REM********************************
```

```
10 REM DEMONSTRATION 1
20 REM
30 REM LOW RESOLUTION PLOTTING
40 REM
50 SA = 49152: COLOUR = 1: TYPE = 1: PRINTCHR$(147)
60 FOR X2 = 0 TO 79 STEP 2
70 SYS SA+3,39,0,X2,49,COLOUR,TYPE
80 NEXT X2
90 COLOUR = 3
100 FOR X2 = 0 TO 79 STEP 2
110 SYS SA+3,39,49,X2,0,COLOUR,TYPE
120 NEXT X2
130 COLOUR = 5:FOR Y = 0 TO 49
140 DN = RND(1)*20
150 FOR I = 1 TO DN
160 SYS SA, RND(1)*80,Y,COLOUR,TYPE
170 NEXT I
180 NEXT Y
190 FOR D = 1 TO 1000 :NEXT
200 RUN
```





```
10 LISTING 2
20 REM
30 REM
40 DATA76,15,195,76,202,195,76,67,196,76,100,196,76,107,196,32,253,174,32,138
50 DATA173,32,247,183,165,20,240,13,169,224,141,131,3,169,216,141,130,3,76
60 DATA51,195,169,216,141,131,3,169,208,141,130,3,169,0,133,253,173,130,3,133
70 DATA254,169,0,133,251,169,8,133,252,173,14,220,41,254,141,14,220,165,1,41
80 DATA251,133,1,160,0,177,253,145,251,165,251,24,105,1,133,251,165,252,105
90 DATA0,133,252,24,165,253,105,1,133,253,165,254,105,0,133,254,165,253,208
100 DATA222,165,254,205,131,3,208,215,165,1,9,4,133,1,173,14,220,9,1,141,14
110 DATA220,169,18,141,24,208,173,232,3,201,255,208,1,96,169,160,160,195,32
120 DATA30,171,96,147,42,42,32,82,69,45,68,69,70,73,78,65,66,76,69,32,67,72
130 DATA65,82,65,67,84,69,82,32,83,69,84,32,69,78,65,66,76,69,68,32,42,42,0
140 DATA169,91,32,255,174,32,138,173,32,247,183,165,20,133,251,165,21,240,44
150 DATA169,229,160,195,32,30,171,96,63,32,67,72,65,82,65,67,84,69,82,32,78
160 DATA85,77,66,69,82,32,71,82,69,65,84,69,82,32,84,72,65,78,32,50,53,53,0
170 DATA169,0,133,252,169,93,32,255,174,6,251,38,252,6,251,38,252,6,251,38
180 DATA252,24,165,252,105,8,133,252,169,0,141,232,3,32,253,174,32,138,173
190 DATA32,247,183,172,232,3,165,20,145,251,200,238,232,3,192,8,208,232,96
200 DATA32,253,174,32,138,173,32,247,183,165,20,201,255,208,9,169,131,141,119
210 DATA2,169,1,133,198,169,16,141,130,2,32,248,252,96,169,18,141,24,208,96
220 DATA96,169,21,141,24,208,96
230 T=0:FORI=49920 TO 50288
240 READ X: POKEI,X: T=T+X:NEXT
250 IF T<>43474 THEN PRINT"ERROR IN DATA"
260 REM********************************
270 REM*   USER DEFINED CHARACTERS  *
280 REM*                            *
290 REM* THIS OCCUPIES $C300-$C470  *
300 REM*                            *
310 REM********************************
```

```
10 REM DEMONSTRATION 2
20 REM
30 REM USER DEFINED CHARACTERS
40 REM
50 REM *********** WARNING ***********
60 REM RAISE THE START OF BASIC BEFORE
70 REM       RUNNING THIS PROGRAM
80 REM ******************************
90 REM
100 POKE1000,255:REM NO MESSAGE
110 POKE53280,1:POKE53281,1:REM WHITE SCREEN
120 SA=49920:REM START OF ROUTINES
130 SYS SA,0:REM UPPER CASE SET
140 GOSUB 310
150 GOSUB450 : REM FILL SCREEN WITH BLANKS (CHARACTER 32)
160 SYS SA+3 [32],255,129,129,129,129,129,129,255:GOSUB440
170 SYS SA+3 [32],255,255,195,195,195,195,255,255:GOSUB440
180 SYS SA+3 [32],255,255,255,231,231,255,255,255:GOSUB440
190 SYS SA+3 [32],255,255,255,255,255,255,255,255:GOSUB440
200 C=C+1:IFC<20THEN160
210 SYS 49923 [32],0,0,0,0,0,0,0,0
220 GOSUB450:FORI=1TO8:READ Y(I):NEXT
230 FORI=1TO8
240 FORJ=1TO8:X(J)=Y(I):NEXT
250 X(I)=255
260 SYS SA+3 [32],X(1),X(2),X(3),X(4),X(5),X(6),X(7),X(8)
270 NEXT:C=C+1:IFC<30THEN230
280 RUN
290 DATA 7,1,4,8,2,6,3,5
300 DATA 128,64,32,16,8,4,2,1
310 FORI=1TO26: SYS SA+3 [I],0,0,0,0,0,0,0,0 : NEXT:GOSUB450
320 PRINT"QQQQQQQQQQ"TAB(3)"A SHORT DEMONSTRATION SHOWING WHAT"
330 PRINT"QQ"TAB(3)"CAN BE ACHIEVED WITH USER DEFINED"
340 PRINT"QQ"TAB(15)"CHARACTERS"
350 POKE56334,PEEK(56334)AND254
360 POKE1,PEEK(1)AND251
370 FOR I=1TO8: READ K
380 FOR J=1TO27
390 POKE 2048+(J-1)*8+K,PEEK(53248+(J-1)*8+K):NEXTJ,I
400 POKE1,PEEK(1)OR4
410 POKE56334,PEEK(56334)OR1
420 FORD=1TO100:Z=RND(1)*256 :SYS 49923 [32],Z,Z,Z,Z,Z,Z,Z,Z:NEXT
430 SYS 49923 [32],0,0,0,0,0,0,0,0:RETURN
440 FORI=1TO20:NEXT:RETURN:REM DELAY
450 O$="":FORI=1TO38:O$=O$+CHR$(32):NEXT
460 PRINT"♥":FORI=1TO23:PRINT"█"O$:NEXT:PRINT"█":RETURN
```

```
10 LISTING 3
20 REM
30 REM
40 DATA32,253,174,32,138,173,32,247,183,165,20,141,134,3,32,253,174,32,138
50 DATA173,32,247,183,165,20,141,135,3,32,253,174,32,138,173,32,247,183,165
60 DATA20,141,132,3,32,253,174,32,138,173,32,247,183,165,20,141,134,2,173,132
70 DATA3,201,91,16,47,201,47,48,43,56,233,48,141,132,3,174,135,3,172,134,3
80 DATA24,32,240,255,172,132,3,185,111,197,133,251,185,154,197,133,252,160
90 DATA0,177,251,201,64,240,6,32,210,255,200,208,244,96,146,179,194,215,240
100 DATA13,40,73,90,123,156,157,158,159,160,161,162,197,228,7,36,67,90,109,140
110 DATA174,193,212,235,250,29,64,95,121,152,182,209,227,4,35,68,99,121,200
120 DATA200,200,200,200,201,201,201,201,201,201,201,201,201,201,201,201,197
130 DATA197,198,198,198,198,198,198,198,198,198,198,198,199,199,199,199,199
140 DATA199,199,199,200,200,200,200,200,172,18,162,146,187,17,157,157,157,161
150 DATA32,18,161,17,157,157,157,172,162,187,17,157,157,157,146,161,32,18,161
160 DATA146,64,18,172,162,146,191,17,157,157,157,18,188,146,162,18,191,17,157
170 DATA157,157,146,161,32,18,161,17,157,157,157,188,146,162,18,191,146,64
180 DATA172,18,162,146,191,17,157,157,157,161,32,32,17,157,157,157,161,32,32
190 DATA17,157,157,157,188,162,18,191,146,64,18,172,162,146,187,17,157,157
200 DATA157,161,32,18,161,17,157,157,157,146,161,32,18,161,17,157,157,157,188
210 DATA146,162,190,64,18,172,162,162,17,157,157,157,188,146,162,17,157,157
220 DATA161,17,157,18,188,146,162,162,64,18,172,162,162,17,157,157,157,188
230 DATA146,162,17,157,157,161,17,157,161,64,172,18,162,146,191,17,157,157
240 DATA157,161,32,32,17,157,157,157,161,32,18,187,146,17,157,157,157,188,162
250 DATA18,191,146,64,161,32,18,161,17,157,157,157,18,188,146,162,18,190,17
260 DATA157,157,157,146,161,32,18,161,146,17,157,157,157,161,32,18,161,146
270 DATA64,29,188,18,172,146,17,157,161,17,157,161,17,157,157,172,18,188,146
280 DATA64,29,188,18,172,17,157,146,161,17,157,161,17,157,157,157,191,162,190
290 DATA64,161,32,18,191,17,157,157,157,188,191,17,157,157,172,146,191,17,157
300 DATA157,161,32,191,64,161,17,157,161,17,157,161,17,157,18,188,146,162,162
310 DATA64,18,188,146,32,18,190,17,157,157,157,146,161,18,162,161,17,157,157
320 DATA157,146,161,32,18,161,17,157,157,157,146,161,32,18,161,146,64,18,188
330 DATA146,32,18,161,146,17,157,157,157,161,161,18,161,146,17,157,157,157
340 DATA161,18,161,161,146,17,157,157,157,161,32,18,187,146,64,172,18,162,146
350 DATA187,17,157,157,157,161,32,18,161,146,17,157,157,157,161,32,18,161,146
360 DATA17,157,157,157,188,162,190,64,18,172,162,146,191,17,157,157,157,18
370 DATA188,146,162,18,191,17,157,157,157,146,161,17,157,146,161,64,172,18
380 DATA162,146,187,17,157,157,157,161,32,18,161,146,17,157,157,157,161,172
390 DATA18,161,146,17,157,157,157,188,162,191,64,18,172,162,146,191,17,157
400 DATA157,157,18,188,146,162,18,191,17,157,157,157,146,161,191,17,157,157
410 DATA146,161,32,191,64,18,191,162,146,191,17,157,157,157,191,187,17,157
420 DATA157,32,188,191,17,157,157,157,191,162,18,191,146,64,188,18,187,162
430 DATA17,157,157,146,18,161,17,157,161,17,157,161,146,64,161,32,18,161,146
440 DATA17,157,157,157,161,32,18,161,146,17,157,157,157,161,32,18,161,146,17
450 DATA157,157,157,191,162,18,191,146,64,161,32,18,161,146,17,157,157,157
460 DATA161,32,18,161,146,17,157,157,157,161,32,18,161,146,17,157,157,157,188
470 DATA162,190,64,161,32,18,161,146,17,157,157,157,161,32,18,161,146,17,157
480 DATA157,157,161,162,18,161,17,157,157,157,172,146,32,18,187,146,64,161
490 DATA32,18,161,146,17,157,157,157,188,162,190,17,157,157,157,172,18,162
500 DATA146,187,17,157,157,157,161,32,18,161,146,64,18,161,146,32,18,161,146
510 DATA17,157,157,191,190,17,157,18,157,161,17,157,161,146,64,18,162,162,187
520 DATA146,17,157,157,172,190,17,157,157,157,172,190,17,157,157,18,188,146
530 DATA162,162,64,18,191,162,146,191,17,157,157,157,161,29,18,161,17,157,157
540 DATA157,146,161,29,18,161,17,157,157,157,146,191,162,18,191,146,64,32,18
550 DATA190,17,157,161,17,157,161,17,157,190,146,187,64,18,191,162,146,191
560 DATA17,157,18,191,17,157,157,191,17,157,157,190,146,162,162,64,18,191,162
570 DATA146,191,17,157,157,162,18,191,17,157,161,17,157,157,157,146,191,162
580 DATA18,191,146,64,29,18,191,146,161,17,157,157,157,18,191,29,146,161,17
590 DATA157,157,157,18,188,146,162,18,188,17,157,146,161,64,18,172,162,162
600 DATA17,157,157,157,188,146,162,187,17,157,18,161,17,157,157,157,146,191
610 DATA162,18,191,146,64,18,191,162,146,191,17,157,157,157,18,188,146,162
620 DATA187,17,157,157,157,161,29,18,161,17,157,157,157,146,191,162,18,191
630 DATA146,64,18,161,162,187,17,157,191,17,157,157,18,161,17,157,161,146,64
640 DATA18,191,162,146,191,17,157,157,157,191,162,18,191,17,157,157,157,146
650 DATA161,29,18,161,17,157,157,157,146,191,162,18,191,146,64,18,191,162,146
660 DATA191,17,157,157,157,161,29,18,161,17,157,157,157,146,188,18,162,187
670 DATA17,157,157,157,146,191,162,18,191,146,64,64,64,64,64,64,64,64
680 T=0:FOR I=50432 TO 51618
690 READ X: T=T+X: POKE I,X:NEXT
700 IF T<>147899 THEN PRINT"DATA ERROR"
710 REM****************************
720 REM* LARGE 4 BY 4 CHARACTERS *
730 REM*  OCCUPIES $C500-$C9A2   *
740 REM****************************
```

```
10 REM DEMONSTRATION 3
20 REM
30 REM USE OF 4 BY 4 ENLARGED CHARACTERS
40 REM
50 PRINTCHR$(147)
60 ME$="A TEST": Y=0: XS=6: GOSUB 120
70 ME$=" DEMO": Y=5: XS=6: GOSUB 120
80 ME$=" USING": Y=10: XS=3: GOSUB 120
90 ME$=" LARGE": Y=15: XS=3: GOSUB 120
100 ME$="LETTERS": Y=20: XS=3: GOSUB 120
110 COLOUR=RND(1)*16 :PRINT"■";:GOTO60
120 FORI=1TOLEN(ME$)
130 CH=ASC(MID$(ME$,I,1))
140 X=(I-1)*5+XS
150 SYS 50432,X,Y,CH,COLOUR:NEXT
160 RETURN
```

```
10 LISTING 4
20 REM
30 REM
40 DATA169,192,133,176,133,187,169,152,133,180,133,189,169,7,133,177,133,181
50 DATA169,219,133,188,133,190,169,24,141,0,192,96,32,0,202,160,39,177,180
60 DATA145,176,177,189,145,187,136,16,245,206,0,192,48,45,165,180,133,176,165
70 DATA181,133,177,165,189,133,187,165,190,133,188,56,165,180,233,40,133,180
80 DATA165,181,233,0,133,181,56,165,189,233,40,133,189,165,190,233,0,133,190
90 DATA24,144,193,160,39,169,0,133,176,169,4,133,177,169,32,145,176,136,16
100 DATA251,96,160,39,185,192,7,153,62,3,185,192,219,153,122,3,136,16,241,32
110 DATA0,202,160,39,177,180,145,176,177,189,145,187,136,16,245,206,0,192,48
120 DATA45,165,180,133,176,165,181,133,177,165,189,133,187,165,190,133,188,56
130 DATA165,180,233,40,133,180,165,181,233,0,133,181,56,165,189,233,40,133
140 DATA189,165,190,233,0,133,190,24,144,193,160,39,185,62,3,153,0,4,185,122
150 DATA3,153,0,216,136,16,241,96
160 T=0: FOR I= 51712 TO 51926
170 READ X:T=T+X: POKEI,X: NEXT
180 IF T<>29451 THEN PRINT"ERROR IN DATA"
190 REM*********************************
200 REM*     SCROLL AND ROLL DOWN      *
210 REM*                               *
220 REM* THIS OCCUPIES $CA00 TO $CA72 *
230 REM*********************************
```

```
10 LISTING 5
20 REM
30 REM
40 DATA173,133,3,141,132,3,206,132,3,169,0,133,176,133,187,169,4,133,177,169
50 DATA216,133,188,96,32,0,203,160,38,177,176,200,145,176,136,177,187,200,145
60 DATA187,136,136,16,241,160,0,169,32,145,176,206,132,3,48,25,24,165,176,105
70 DATA40,133,176,144,2,230,177,24,165,187,105,40,133,187,144,208,230,188,24
80 DATA144,203,96,32,0,203,160,39,177,176,141,134,3,177,187,141,135,3,136,177
90 DATA176,200,145,176,136,177,187,200,145,187,136,136,16,241,160,0,173,134
100 DATA3,145,176,173,135,3,145,187,206,132,3,48,25,24,165,176,105,40,133,176
110 DATA144,2,230,177,24,165,187,105,40,133,187,144,191,230,188,24,144,186,96
120 DATA32,0,203,160,1,177,187,136,145,187,200,177,176,136,145,176,200,200,192
130 DATA40,208,239,160,39,169,32,145,176,206,132,3,48,25,24,165,176,105,40
140 DATA133,176,144,2,230,177,24,165,187,105,40,133,187,144,206,230,188,24
150 DATA144,201,96,32,0,203,160,0,177,176,141,134,3,177,187,141,135,3,200,177
160 DATA187,136,145,187,200,177,176,136,145,176,200,200,192,40,208,239,160
170 DATA39,173,134,3,145,176,173,135,3,145,187,206,132,3,48,25,24,165,176,105
180 DATA40,133,176,144,2,230,177,24,165,187,105,40,133,187,144,189,230,188
190 DATA24,144,184,96
200 T=0: FORI=51968 TO 52257
210 READ X: POKE I,X: T=T+X: NEXT
220 IF T <> 37953 THEN PRINT"ERROR IN DATA"
230 REM*********************************
240 REM*  SCROLL/ROLL LEFT AND RIGHT  *
250 REM*                               *
260 REM* THIS OCCUPIES $CB00 TO $CC21 *
270 REM*********************************
```

```
10 REM DEMONSTRATION 4
20 REM SCROLLS AND ROLLS
30 REM
40 PRINTCHR$(147)
50 PRINT"▤*********"
60 PRINT"▤*********"
70 PRINT"▤*********"
80 PRINT"■*********"
90 PRINT"■*********"
100 PRINT"■*********"
110 FORI=1TO8:SYS 51742: NEXT
120 POKE 901,24: FORI=1TO96: SYS 52049: NEXT
130 POKE 901,12: FORI=1TO40: SYS 52049: NEXT
140 POKE 901,24: FORI=1TO19: SYS 51992: NEXT
150 POKE 901,24: FORI=1TO37: SYS 52123: NEXT
160 FORI=1TO18: SYS 51992: NEXT
170 FORI=1TO38: SYS 51826: NEXT
180 POKE 901,4: FORI=1TO12: SYS 52182: NEXT
190 POKE 901,25
200 SYS 52182: GOTO 200
```

☐

TONY CROSS & PHIL CORNES

Getting More From The 64

Dig deep into your 64 and explore the mysteries of sprites, hi-res graphics and sound.

There is a saying that a computer is only as good as its manual. Certainly, in the case of the Commodore 64, a very powerful machine is being let down by an inadequate manual. Over the next few pages we hope to remedy some of the problems by describing how to get the best out of the sprite, sound and high resolution graphics facilities on this otherwise excellent machine.

PART 1: CREATING SPRITES

Sprites are user-definable high resolution graphics characters. They are displayed on a grid 24 dots wide by 21 dots high, with each dot on the grid controlling a dot on the high resolution screen. The sprites themselves are controlled by the Video Interface Chip, or VIC chip for short. To create a sprite we have to tell the VIC chip which dots to light and which dots not to light to produce the shape we require.

The easiest way to do this is to begin by drawing the shape of the sprite on some squared paper which is 24 squares wide by 21 squares high. Each square represents a dot on the high resolution screen. By filling in those squares which must be lit, it is very easy to create the shape you require.

To enable this shape to be stored in the computer each 'filled in' square should be given the value 1 and each 'unfilled' square the value 0. By grouping the squares (or bits) together in groups of eight, we can form a 3 byte by 21 byte grid (63 bytes in all). These 63 bytes can now be stored as a contiguous 63 byte block of memory.

STORING SPRITE DATA

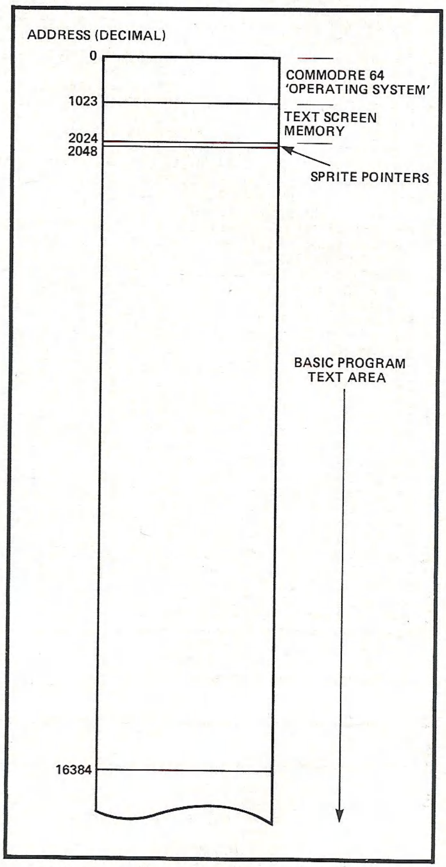
The problem now is to find a 'safe' place in memory to store the sprite data. The question is . . . where? Well, let's look at the problem from the VIC chip's point of view.

The VIC chip can control up to eight sprites from BASIC, each with different data. To sort out where the data for each sprite is located it looks at a group of registers called the sprite data pointers. These are at locations 2040 decimal to 2047 decimal. Each register points to the data for one sprite. For example, 2040 points to the data for sprite 0, 2041 points to sprite 1's data and so on.

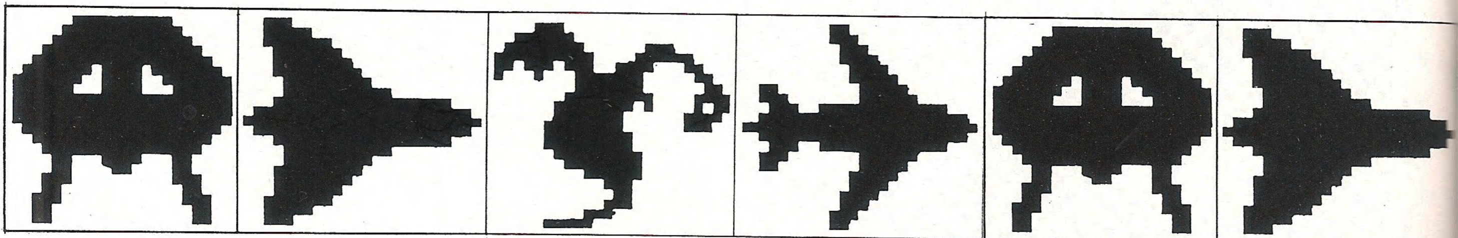
The sprite data pointers are all one byte wide: this means that they can each point to 256 different locations. As we have seen it takes 63 bytes to define a sprite. However, for a computer, the number 64 is much more convenient ($64 = 2^6$). For this reason the VIC chip assumes that each sprite's data occupies a 64-byte block of memory (the last byte is not used). When the VIC chip wants to know where a given sprite's data is located it multiplies the value in the sprite data pointer by 64 to give the address of the first byte of data. For example, if register 2040 contained 15, then the VIC chop would look at memory location 960 for the start of sprite 0's data, since $960 = 15 \times 64$.

From this we can work out the range of memory locations which can be used to store sprite data. Since the sprite data pointers can point to 256 locations and sprites take up 64 bytes each then the range is 0 to 16320, in 64-byte blocks, as $16320 = 255 \times 64$. (It is possible to make the sprite data pointers point to locations outside this range, but that is beyond the scope of this article).

Within the area of RAM available for sprite storage, there are many locations where it is not 'safe' to store sprite data. Figure 1 shows the general usage of the first 16K of RAM. The first 1K of RAM is used by the Commodore 64 operating system. The only 'useable' space in here is from location 828 decimal to 1023 decimal. This is actually the cassette I/O buffer but, provided you are not using the cassette recorder within your program, you can use the buffer space to store the data for up to three different sprites. (This is where the demonstration sprites in the User Manual have been located.)



The only other available space is in the BASIC program text area. BASIC text starts at location 2048 decimal and extends up to 40959 decimal, giving you 38911 bytes for BASIC programs. The problem with using this area is that BASIC may overwrite your sprite data. In addition, there is one area within the BASIC text area that cannot be used for storing sprites. This is the block



from 4096 decimal to 8191 decimal. The VIC chip maintains an image of the character set in this area and although BASIC text can be stored here, sprite data cannot.

It is possible to create a 'safe' storage area for sprites by moving the bottom of the BASIC text area up a few bytes. This is not easy to do from BASIC, so we have included a machine code routine at the end of this section which moves the bottom of BASIC up by 512 bytes. This creates enough room to store all eight sprites. The routine, which we have called MEMSHIFT, is called by a SYS 49329 command from BASIC (MEMSHIFT is located at address 49329). The bottom of the BASIC text area will then be moved from 2048 decimal to 2560 decimal. Pressing SHIFT and RESTORE will restore BASIC to normal. Changing the start of the BASIC text area like this will corrupt any programs which are stored in RAM, so MEMSHIFT also calls the BASIC 'NEW' routine to restore BASIC to a sensible state.

Sprites can now be safely stored in locations 2048 decimal to 2559 decimal and any BASIC programs entered from the keyboard or loaded from tape will function normally.

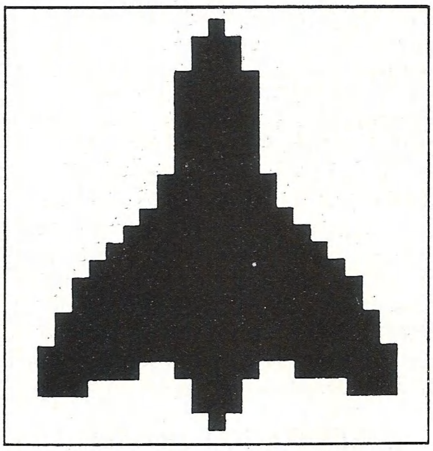
DISPLAYING THE SPRITES

Each sprite can only be in one of two states; it is either being displayed or it is not. These two states can be represented by a 1 for 'on' and a 0 for 'off', which means that all eight sprites can be controlled by one register. This register is called the sprite enable register and it is located at 53269 decimal. Each bit within the register controls one sprite. Bit 0 controls sprite 0, bit 1 controls sprite 1 and so on. For example, to turn sprite 3 on we need to set bit 3 of the sprite enable register to 1. Since we may not know which other sprites are already on or off, the safe way to set bit 3 is POKE 53269,PEEK (53269) OR 4.

POSITIONING SPRITES

Having turned a sprite on we want to be able to position it on the screen. As we mentioned earlier, sprites are displayed on the 320 by 200 high resolution screen, so a pair of X and Y coordinates, within these ranges, is all that is required.

The 16 registers from 53248 decimal to 53263 decimal are used to store the X and Y coordinates of the eight sprites. Figure 2, shows how these registers are organised. The value held in a sprite's X and Y registers gives the position where the top left hand corner of the 24 by 21 dots grid will be placed on the high resolution screen.



However, the screen layout for sprite positioning is not numbered in the way you might expect. Instead of being numbered from 0 to 319 and from 0 to 199 the visible screen area is actually numbered from 24 to 344 in the X direction and 50 to 250 in the Y direction. The sprite screen numbering has been done in this way so that sprites can be smoothly moved on and off the screen in both the X and Y directions without requiring negative coordinates. For example, a sprite positioned at 12, 50 will be fully on screen in the Y direction and half on the screen in the X direction.

In addition, a problem occurs when you are using large values of X. This is because the X registers can only hold numbers up to 255, which is not enough to completely address the 320-dots-wide screen. This has been overcome by using an additional register to provide each X register with an extra bit, thus making them all nine bits wide. (A nine-bit register can hold numbers up to 511). This additional register is called the most significant bit X register (MSBX) and it is located at 53264 decimal. Within the register bit 0 is the extra bit for sprite 0, bit 1 is the extra bit for sprite 1 and so on.

SPRITE NUMBER	X REGISTER	Y REGISTER
0	53248	53249
1	53250	53251
2	53252	53253
3	53254	53255
4	53256	53257
5	53258	53259
6	53260	53261
7	53262	53263

This creates a little bit of extra work for us because we now have to check every X coordinate to see if it is larger or smaller than 255. If it is larger then we must set the extra bit, subtract 256 from the X coordinate and store the result in the X register. If it is smaller, however, we must reset the extra bit and store the whole X coordinate in the X register.

DOT PAIR	DOT COLOUR
00	Transparent
01	Sprite multi-colour register 0
10	Sprite colour register
11	Sprite multi-colour register 1

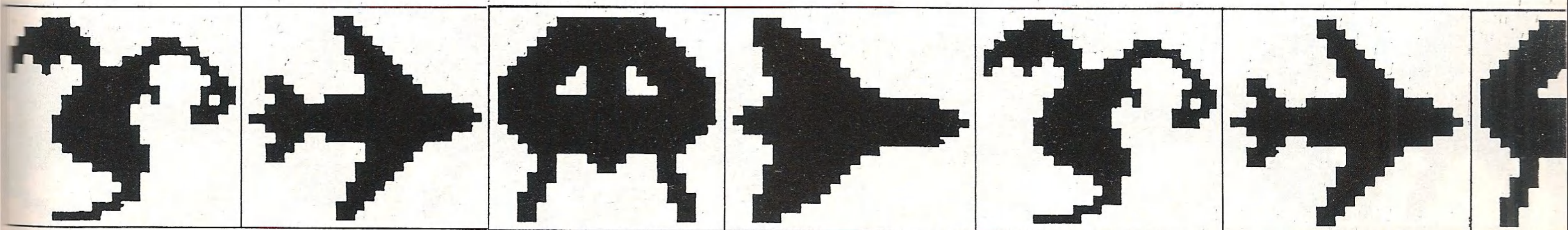
To make this job a little bit easier we have included another machine code routine called SPRITEPOS which will position any of the sprites anywhere on screen. SPRITEPOS uses three BASIC integer variables, SN%, X% and Y%. SN% tells the routine which sprite to move and X% and Y% tell the routine where to put it on the screen. To use SPRITEPOS, set up the integer variables SN%, X% and Y% and then use the SYS 49217 command (SPRITEPOS is located at address 49217).

If a SYNTAX ERROR message is returned by SPRITEPOS it means that one of the three variables was missing. (Remember, they *must* be integer variables.) If you get an ILLEGAL QUANTITY ERROR message it means that SN% was outside the range 0-7.

USING COLOUR

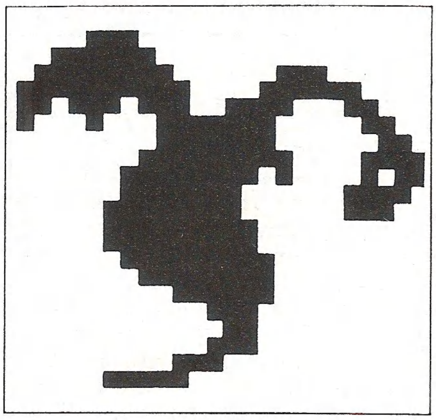
The sprites can be displayed in any of the 16 available colours and in one or two 'modes'. These 'modes' are:

- The standard colour mode, where each sprite can be in one of the 16 colours on a 'transparent' grid.
- Multi-colour mode, where each sprite can be in three different colours on a 'transparent' grid.



The term 'transparent grid' means that any dots on the 24 by 21 dots sprite grid which are set to 0 will allow whatever is 'underneath' them to show through.

In standard colour mode the sprite colour is selected by POKEing the colour value (0 to 15) into the appropriate sprite colour register. These are at locations 53287 decimal to 53294 decimal. There is one register for each sprite; register 53287 is for sprite 0, register 53288 is for sprite 1 and so on, but only the low four bits are effective (four bits can take 16 different states). For example, to display sprite 5 in orange you should POKE 53929,8 (the colour value for orange is 8).



In multi-colour mode you sacrifice horizontal resolution for increased colours. Instead of treating every dot independently as either coloured or transparent, the horizontal dots are now 'read' in pairs so that each pair of dots can be in one of three colours or transparent. Figure 3 shows how the dot pairs are coded into colours. The two sprite multi-colour registers hold the values of the two extra colours, and they are the same for all sprites in the multi-colour mode. these registers are at locations 53285 decimal for register 0 and 52386 decimal for register 1.

To select multi-colour mode for a sprite you must set the appropriate bit in the sprite multi-colour register. This is at location 53276 decimal and, as before, bit 0 is for sprite 0, bit 1 is for sprite 1 and so on. to return to standard colour mode simply reset the appropriate bit back to 0. For example, to select multi-colour mode for sprite 2 you should POKE 53276,PEEK (53276) OR 4.

EXPANDED SPRITES

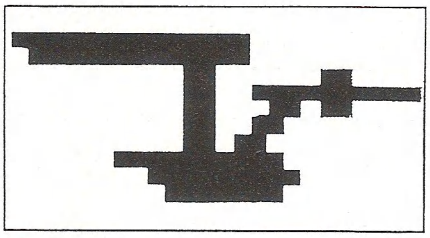
Another facility which can be used to great effect is the sprite expansion facility. Sprites can be doubled in size, in both the X and Y directions, by setting the appropriate bits in the X and/or Y expansion registers. The expansion registers are at locations 53277 for the X direction and 53271 for the Y direction. Once again, bit 0 controls sprite 0, bit 1 sprite 1 and so on in each register. For example, to display sprite 3 expanded in the X direction we must POKE 53277,PEEK (53277) OR 8.

What actually happens is that with expand on, every dot on the 24 by 21 sprite grid occupies two dots on the high resolution screen instead of one. The sprite's resolution does not change; it is still 24 by 21 dots even though with expand on in both directions, the sprite occupies 48 by 42 dots on the screen.

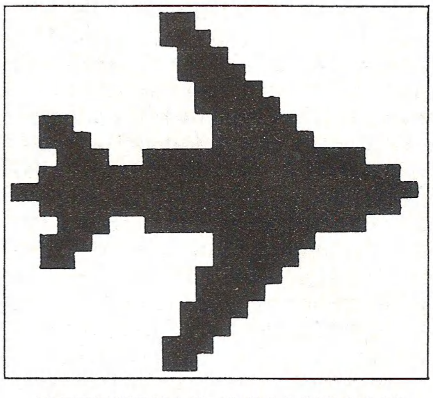
SPRITE PRIORITIES

One of the more powerful facilities of sprites is their ability to be assigned priorities. these priorities determine which sprite will be 'in front' when two sprites move over each other and whether a sprite will pass 'in front of' or 'behind' the background graphics.

The sprite-to-sprite priorities are 'built-in' to the system. Sprite 0 has the highest priority, so no other sprites can appear to apss in front of sprite 0. Sprite 1 has the next highest, so it will pass in front of sprites 2 to 7 but behind sprite 0. The other sprite priorities follow the same pattern.



The sprite-to-background priorities can be individually selected for each of the sprites. This is done by setting the appropriate bit in the sprite background priority register, which is located at 53275 decimal. Again, bit 0 controls sprite 0, bit 1 controls sprite 1 and so on. If the appropriate bit is a 0 then that sprite will pass in front of all background graphics. If the bit is a 1 then that sprite will pass behind all background graphics. For example, to make sprite 4 pass behind the background graphics POKE 53275, PEEK(53275) OR 16.

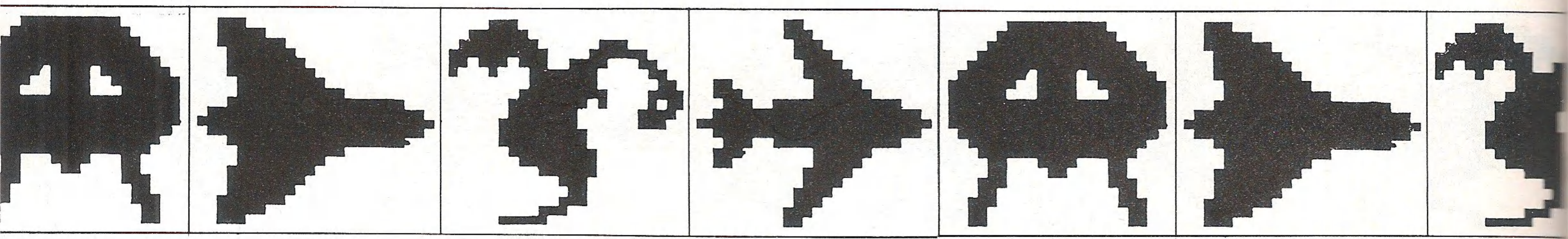


DETECTING COLLISIONS

Finally we come to the most powerful facility of all, the collision detections. The VIC chip can detect when any number of sprites collide, or when any number of sprites collide with background graphics. A collision has occurred when any non-transparent areas of two sprites overlap each other, or when any non-transparent area of a sprite overlaps any background graphics.

The VIC chip reports these collisions back to us via two collision registers. These are the sprite-to-sprite collision register at location 53278 decimal, and the sprite-to-background collison register at location 53279 decimal. As before, bit 0 refers to sprite 0, bit 1 refers to sprite 1 and so on for each register.

For each of the registers the VIC chip sets the appropriate bits to 1 for those sprites which have been involved in collisions. For example, if sprites 0 and 4 have collided with each other register 53278 will contain the value 17 (bits 0 and 4 set). In both cases the collision registers retain the most recent collision information until they have been read by a PEEK. The action of PEEKing the registers automaically resets them, so if you need to retain the values in the registers you must PEEK them into variables.



THE MACHINE CODE ROUTINES

To finish this section see the BASIC program to load both MEMSHIFT and SPRITEPOS into the 4K of user RAM starting at 49152 decimal. (Later there will be some more machine code routines which will add on to MEMSHIFT and SPRITEPOS giving you some fairly powerful graphics routines.) These require variable space in page zero. Unfortunately, Commodore have used nearly all of page zero locations for various system functions. So in order to create space for the variables in our routines we have 'borrowed' the locations used by the KERNAL RS-232 routines. this means that if you are using the RS-232 port you won't be able to use our routines. Sorry about that . . . !

```
10 REM Routine to load MEMSHIFT and SPRITEPOS.
20 FOR A=0 TO 197
30 READ Q : POKE 49152+A,Q
40 NEXT A
100 DATA 165,46,133,254,165,45,133,253
110 DATA 160,0,165,254,197,48,208,8
120 DATA 165,253,197,47,208,2,56,96
130 DATA 177,253,197,252,208,17,200,177
140 DATA 253,197,251,208,11,200,177,253
150 DATA 170,200,177,253,168,24,96,200
160 DATA 152,24,105,6,101,253,133,253
170 DATA 165,254,105,0,133,254,76,8
180 DATA 192,169,211,133,252,169,206,133
190 DATA 251,32,0,192,176,89,152,48
200 DATA 91,201,8,176,87,133,249,10
210 DATA 133,250,169,1,200,136,240,4
220 DATA 10,76,93,192,133,248,169,217
230 DATA 133,252,169,128,133,251,32,0
240 DATA 192,176,52,166,250,152,157,1
250 DATA 208,169,216,133,252,169,128,133
260 DATA 251,32,0,192,176,33,138,72
270 DATA 166,250,152,157,0,208,104,201
280 DATA 0,208,11,165,248,73,255,45
290 DATA 16,208,141,16,208,96,165,248
300 DATA 13,16,208,141,16,208,96,162
310 DATA 11,108,0,3,162,14,108,0
320 DATA 3,162,10,160,0,24,32,156
330 DATA 255,200,132,43,134,44,169,0
340 DATA 141,0,10,76,66,166
```

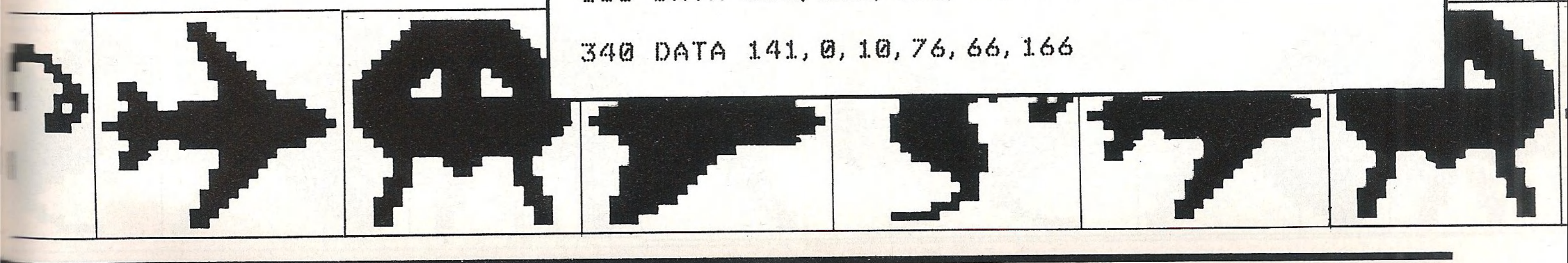
PART 2: HI-RES GRAPHICS

A high resolution graphics screen is now an essential part of any self-respecting micro. The extra resolution means that each screenfull contains more information. Diagrams and graphs can be made more detailed and pictures become more realistic. On the Commodore 64 the 320 by 200 dots high resolution screen is more than adequate for most purposes. However, accessing and using the screen is by no means straightforwared.

SCREEN ORGANISATION

Let's begin by looking at the way the high resolution screen is organised. The high resolution screen is 'bit mapped' into memory. this means that each dot on the screen is controlled by a bit in memory. If the bit is a 1 then the dot will be lit, if it is a 0 then the dot will not be lit.

Armed with this information we can work out how much memory will be needed to store the high resolution screen. The number of dots on the screen is 320 × 200 = 64000. Each byte can control eight dots, so we will need 6400/8 = 8000 bytes to store the screen. In the 64K of RAM available there are eight different places where the screen could be located (65535/8000 = 8.192). However, the VIC chip can



only 'see' 16K of RAM at a time and in the standard configuration it is looking at the first 16K. (It is possible to make the VIC chip look at the other 16K banks of memory, but that's beyond the scope of this article.)

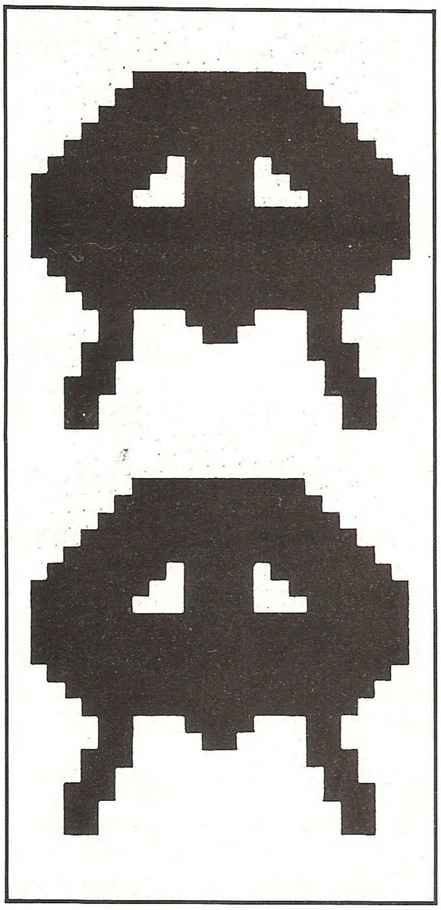
This means that there are only two possible locations for the screen in the standard configuration. These are at location 0 and at location 8192 decimal. (8192 is used because this is 8K or 21^ 13, which is a much more convenient number to a computer than 8000.)

We mentioned above that the first 1K bytes of memory are fully used by the Commodore 64 'operating system'. There are also many BASIC pointers and variables stored in other low memory locations. All this means that the high resolution screen cannot be stored at location 0. So, in the standard configuration, the only place where the high resolution screen can be located is at locations 8192 decimal to 16191 decimal.

LOCATING THE HIGH RESOLUTION SCREEN

To find out where the high resolution screen is located, the VIC chip looks at bit 3 of the VIC memory control register, which is at location 3272 decimal. If the bit is a 0 then the screen is at location 0 (within the current 16K 'bank'). If the bit is a 1 then the screen is at location 8192 (within the current 16K 'bank'). For example, to locate the high resolution screen at 8192 decimal, type POKE 53272, PEEK (53272) OR 8.

Just as with sprite data, locating the high resolution screen at 8192 takes up BASIC text space, and since BASIC starts at location 2048 it is quite likely that BASIC programs will overwrite the high resolution screen. This can be avoided by using a modified version of the MEMSHIFT routine. MEMSHIFT currently moves the bottom of BASIC up to location 2560 decimal to give room for sprite data. By changing a couple of bytes in the routine it can be made to move BASIC up to location 16384 decimal. (This is the closest to location 16191) decimal that you can get, because the bottom of BASIC can only be shifted in multiples of 256 bytes.)



The BASIC program to load this modified version of MEMSHIFT is included at the end of this section. The modified MEMSHIFT is executed in just the same way as the old version, by using the SYS 49329 command from BASIC.

The net result of moving the bottom of BASIC to location 16384 decimal is that we have 'lost' 14K of BASIC space (although we still have 24K left). On the 'plus' side, however, we have protected the high resolution screen and we have enough room between locations 2048 decimal and 4095 decimal for 32 sprites. (Remember from above that sprites cannot be stored between locations 4096 decimal and 8191 decimal.)

USING THE HIGH RESOLUTION SCREEN

Having located the high resolution screen in memory we can now enter high resolution mode. This is done by setting bit 5 of the VIC control register, which is at location 53265 decimal. For example, to enter high resolution mode, use POKE 53265, PEEK(53265) OR 32.

The VIC chip will now start displaying the contents of the high resolution screen. Initially the screen is full of garbage and this comes from two sources. First, the contents of memory locations 8192 to 1619 are probably full of rubbish and this will be displayed as red dots on a black background. In addition, any text characters that were being displayed on the normal text screen are now displayed as different coloured squares on the high resolution screen. This happens because the normal text screen is used to store the high resolution colour information (more on this later).

The first thing to be done, then, is to clear the high resolution screen. The coloured squares can be removed by printing a 'clear screen' character (Shift/Clr), although they will reappear if you type anything on the keyboard. Clearing the high resolution screen memory can be done from BASIC as follows:

```
100 FOR I=0 to 7999: POKE
8192+I,0: NEXT I
```

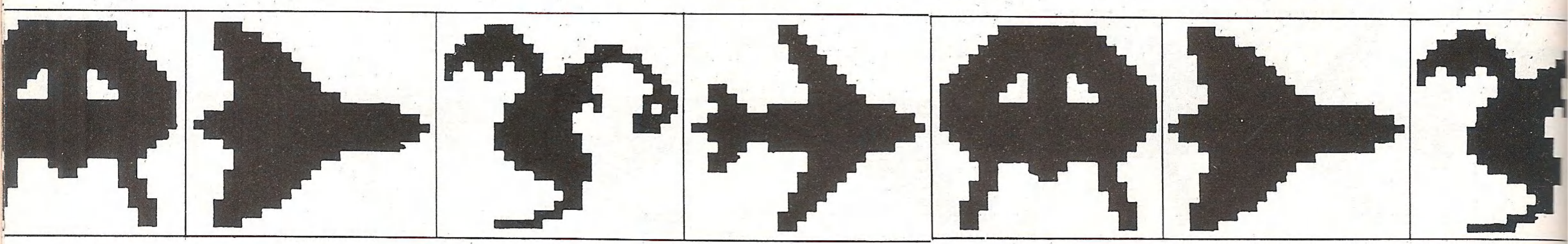
Since this is a very slow process (it takes about 15 seconds to clear the screen), we have also included a machine code clear screen routine. The routine is called CLEAR and it is called by a SYS 49378 command from BASIC. After executing CLEAR, all memory locations from 8192 to 16191 will be set to zero.

PLOTTING POINTS ON THE SCREEN

Just before we actually start drawing on the high resolution screen we need to look at the way in which the screen is laid out. there are two ways of visualising the layout of the high resolution screen. These are:

- The logical view, which is the way you would like the screen to be organised, as shown in Fig. 4.
- The physical view, which is the way the screen is really organised, as shown in Fig. 5.

Obviously the logical view is the one we want to be using when we refer to screen locations, so a conversion formula to convert from the logical to the physical view is needed. There isn't enough space to be



able to describe how the formula is arrived at, but here it is anyway. (Assumed that the x and y coordinates are given by X and Y respectively.) The byte which must be changed is given by:

BYTE = 8192 + INT(Y/8) * 320 + INT(X/8) * 8 + (Y AND 7)

The bit within that byte which must be set is given by:

POKE BYTE, PEEK(BYTE) OR 2 ^ (7 — (X AND 7))

In BASIC this tends to be fairly slow, particularly if you use it as part of a line draw routine. To overcome this problem we have included another machine code routine which does the conversion and lights the appropriate dot for you.

The routine uses the value in the integer variable X% as the x coordinate. There are two 'entry points' to this routine: one called PLOT, which turns on the dot at X%, Y%, and one called UNPLOT which turns off the dot at X%, Y%.

To use the routine, first set up X% and Y% to contain the desired coordinate, then use the SYS 49475 command for the PLOT entry point and the SYS 49488 command for the UNPLOT entry point. No range checking is done on either of the two coordinates although the Y coordinate is always treated as a once byte number, which means that it can never be greater than 255. If either X% or Y% does not exist then a SYNTAX ERROR message will be returned.

USING COLOUR

As we mentioned earlier, the colour information for the high resolution screen comes from the normal text screen at locations 1024 decimal to 2023 decimal. Each text screen location holds the colour for a square eight dots wide by eight dots high on the high resolution screen. Location 1024 controls the colour for screen bytes 0 to 7, location 1025 controls the colour for screen bytes 8 to 15 and so on (see Fig. 5).

Like sprites, the high resolution screen can operate in two colour modes. These are:

- Standard colour mode where each dot on the screen can be one of two colours.
- Extended colour mode where each dot can be in one of four colours.

In 'standard colour' mode the high four bits of the appropriate text screen location hold the colour for a dot which is lit (1), while the low four bits of the text screen location hold the colour for a dot which is not lit (0). Usually the whole screen will be using the same two colours which means that the appropriate composite colour value must be written into the whole text screen. (This is why the screen was initially red dots on a black background. A 'cleared' text screen is really full of 'space' characters which are code 32 decimal. This translates to a red foreground and a black background!)

You can fill the screen with any combination of foreground and background colours by using the following line of code where the variable C contains the composite colour:

```
110 FOR I=1 TO 999: POKE 1024+I,C: NEXT I
```

Like the BASIC clear screen routine this process is also fairly slow. In addition, sorting out the composite colour value can be a bit awkward. To overcome these problems we have included another machine code routine to set up the colours for you.

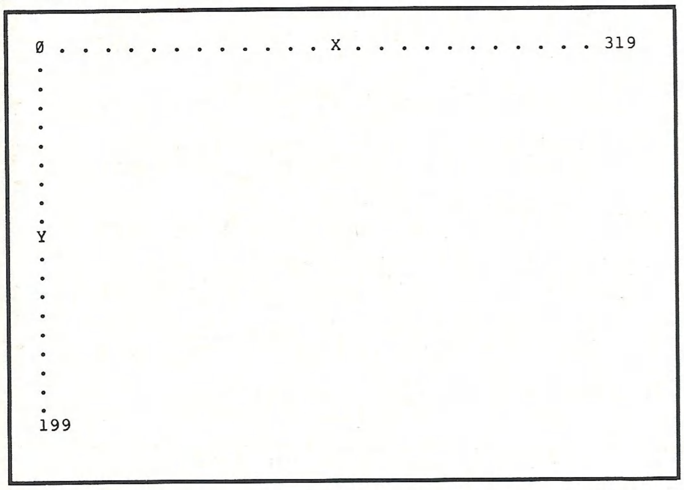
The routine is called COLOUR and it reads the values in the BASIC integer variables FC% and BC% to get the foreground and background colours respectively. To use COLOUR, first set up FC% and BC% and then call COLOUR with the SYS 49401 command from BASIC. The whole of the text screen memory will then be filled with the appropriate composite colour value.

No range checking is done on either of the colours but only the low four bits of each value is 'read' which limits their range to 0-15. If a SYNTAX ERROR Message is returned by COLOUR then either FC% or BC% did not exist.

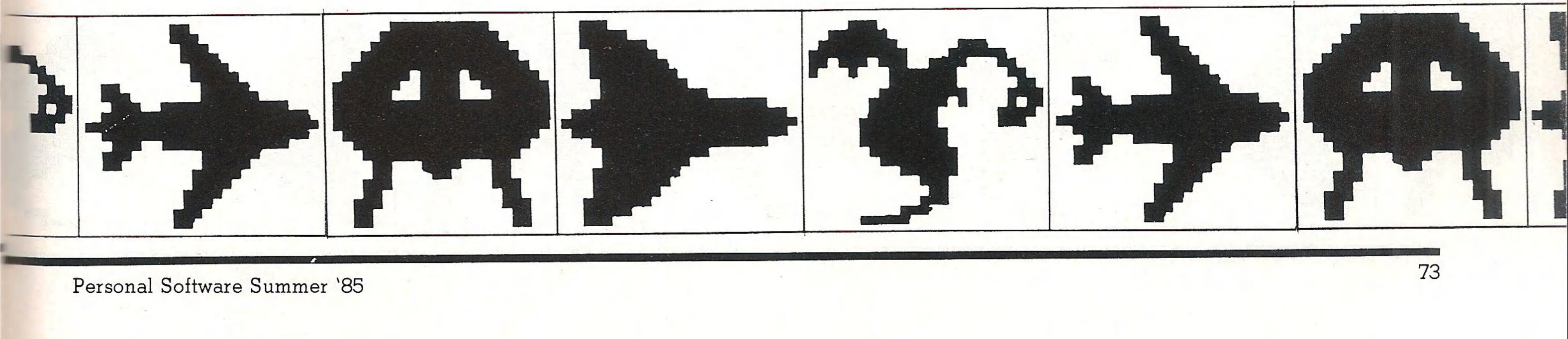
In 'extended colour' mode you sacrifice horizontal resolution for increased colours. Instead of every dot being in one of two colours, the horizontal dots are now 'read' in pairs so that each pair of dots can be in one of four colours. Figure 6 shows how the dot pairs are coded into colours.

The background colour register 0 is located at 53281 decimal and it normally holds the text screen background colour. The colour value in this register is the same for the whole high resolution screen.

The normal colour memory runs from 55296 decimal to 56295 decimal. As with the text screen memory, each colour memory loca-



```
          byte 0    byte 8    byte 16   . . . .   byte 312
          byte 1    byte 9    byte 17   . . . .   byte 313
          byte 2    byte 10   byte 18   . . . .   byte 314
TOP       byte 3    byte 11   byte 19   . . . .   byte 315
LINE      byte 4    byte 12   byte 20   . . . .   byte 316
          byte 5    byte 13   byte 21   . . . .   byte 317
          byte 6    byte 14   byte 22   . . . .   byte 318
          byte 7    byte 15   byte 23   . . . .   byte 319


          byte 320  byte 328  byte 336   . . . .  byte 632
          byte 321  etc
          byte 322  etc
NEXT      byte 323  etc
LINE      byte 324  etc
          byte 325  etc
          byte 326  etc
          byte 327  etc


          etc etc
```



tion controls the colour for an eight-by-eight square.

To select extended colour mode for the high resolution screen you must set bit 4 of the second VIC control register at location 53270 decimal. For example, to select extended colour mode POKE 53270, PEEK(53270) OR 16.

THE MACHINE CODE ROUTINES

Finally, here is the BASIC program to load the modified version of MEMSHIFT and the CLEAR, COLOUR, PLOT and UNPLOT routines. Before loading these routines you must first load listing 1, because these routines call several subroutines which were in listing 1.

PART 3: SOUND

There is no doubt that good sound effects can make even the simplest of games much more exciting to play. But the use of sound on computing need not be restricted to the games applications. Sound, when used properly, can provide as much information as several lines of text. Warning tones, advisory tones, audio feedback on data entry and, on the more advanced machines, real music are all possibilities.

The sound system on the Commodore 64 is well able to provide almost limitless sound effects and excellent music. And because the sound facilities are accessed by POKEing individual registers the system is very flexible. This allows experimentation with all sorts of unlikely combinations of tones and filters when developing sound effects.

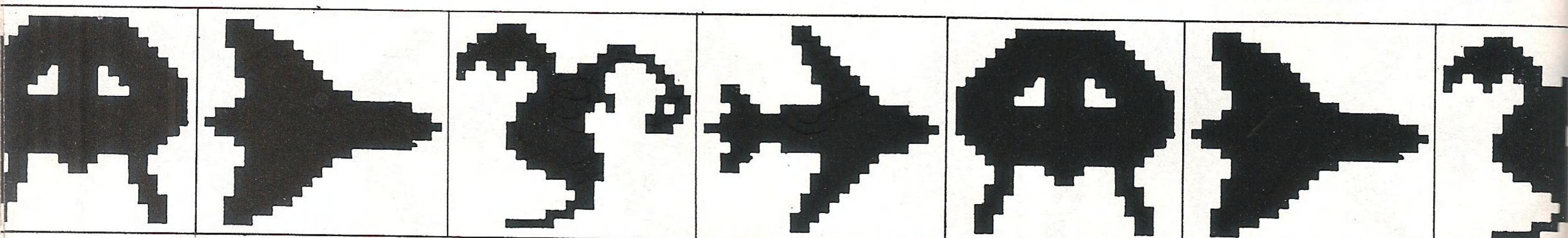
THE SOUND SYSTEM ORGANISATION

The sound system is controlled by the Sound Interface Device or SID chip for short. Internally it has three completely separate sound channels or voices. the voices are organised as shown in Fig. 7.

Each of the sound sources can produce four different waveforms, triangular, sawtooth, rectangular (pulse) and white noise. The frequencies of each of the waveforms can be individually varied and the pulse

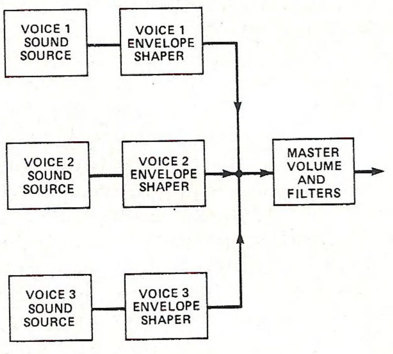
DOT PAIR	DOT COLOUR
00	Background colour register 0
01	High four bits of the text screen memory
10	Low four bits of the text screen memory
11	Normal colour memory

```
10 REM Routine to load MEMSHIFT, CLEAR,
   COLOUR, PLOT & UNPLOT
20 FOR A=0 TO 273

30 READ Q : POKE 49329+A,Q

40 NEXT A

100 DATA 162,64,160,0,24,32,156

110 DATA 255,200,132,43,134,44,169,0

120 DATA 141,0,64,76,66,166,165,168

130 DATA 197,170,208,7,165,167,197,169

140 DATA 208,1,96,160,0,165,171,145

150 DATA 167,230,167,208,233,230,168,76

160 DATA 198,192,169,0,133,167,169,32

170 DATA 133,168,169,64,133,169,169,63

180 DATA 133,170,169,0,133,171,76,198

190 DATA 192,169,195,133,251,169,194,133

200 DATA 252,32,0,192,176,46,152,41

210 DATA 15,133,171,169,195,133,251,169

220 DATA 198,133,252,32,0,192,176,28

230 DATA 152,10,10,10,10,5,171,133

240 DATA 171,169,4,133,168,169,0,133

250 DATA 167,169,7,133,170,169,232,133

260 DATA 169,76,198,192,169,27,141,17

270 DATA 208,169,21,141,24,208,162,11

280 DATA 108,0,3,32,95,193,32,128
```



```
290 DATA 193, 160, 0, 17, 180, 145, 180, 96

300 DATA 32, 95, 193, 32, 128, 193, 160, 0

310 DATA 73, 255, 49, 180, 145, 180, 96, 169

320 DATA 216, 133, 252, 169, 128, 133, 251, 32

330 DATA 0, 192, 176, 200, 134, 181, 132, 180

340 DATA 169, 217, 133, 252, 169, 128, 133, 251

350 DATA 32, 0, 192, 176, 183, 132, 182, 96

360 DATA 165, 180, 72, 41, 7, 133, 183, 104

370 DATA 41, 248, 133, 180, 165, 182, 72, 41

380 DATA 7, 133, 184, 104, 41, 248, 72, 74

390 DATA 74, 74, 133, 182, 74, 74, 24, 101

400 DATA 182, 133, 182, 104, 10, 10, 10, 5

410 DATA 184, 24, 101, 180, 133, 180, 165, 182

420 DATA 101, 181, 24, 105, 32, 133, 181, 169

430 DATA 128, 166, 183, 232, 202, 240, 3, 74

440 DATA 16, 250, 96
```

width ratio of the rectangular waveform can also be varied. Figure 8 shows the shapes of the four different waveforms. For any particular note, these four waveforms produce very different sounds.



The sound is then fed into an envelope shaper where the Sustain level and the Attack, Decay and Release rates can be set up (the ADSR envelope). Figure 9 shows a typical ADSR envelope. It is this envelope shape which gives musical instruments their distinctive sounds and it is possible to 'simulate' some of them quite well. Alternatively new 'computer instrument' sounds can be produced by using other envelope shapes.

Finally the outputs from the three voices are brought together under a master volume control. Three types of tone filtering can also be introduced at this stage. These are a high-pass, a low pass and a band pass filter. the cutoff frequencies of these filters can be individually selected. Figure 10 shows the effects of the three types of filter.

THE SOUND SOURCES

Commodore call the sound sources 'waveform generators' and they have two main functions:

- To produce a note of the selected frequency in the selected waveform.
- To enable the waveform generator output to be turned on or off.

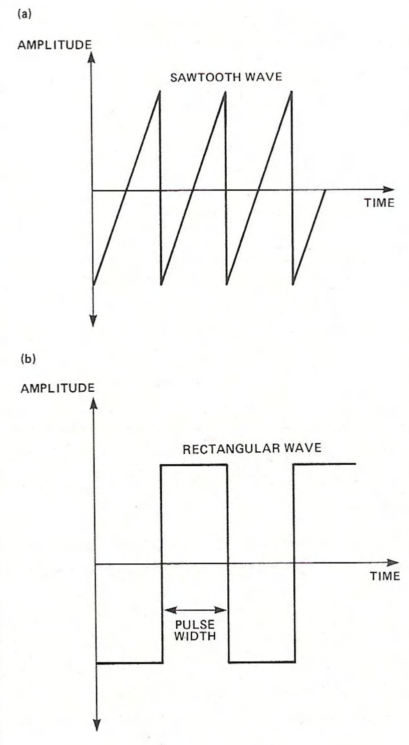
For each voice both these functions are controlled by the same register, the voice control register. These control registers are at the following locations:

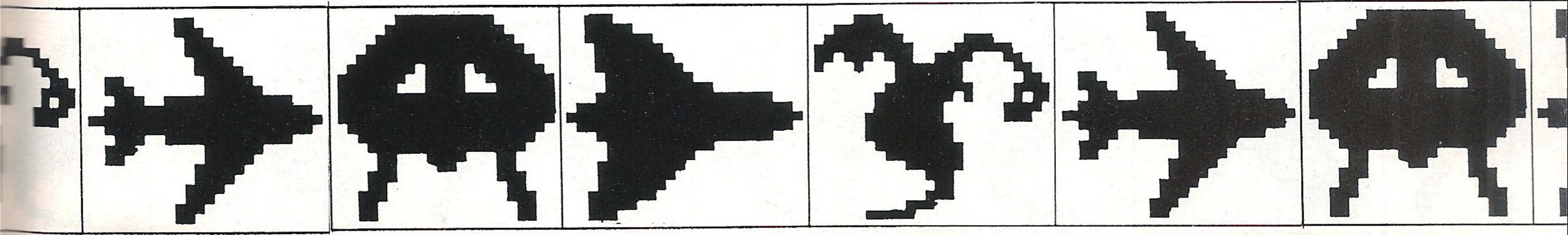
Voice 1=54276
Voice 2=54283 *
Voice 3=54290

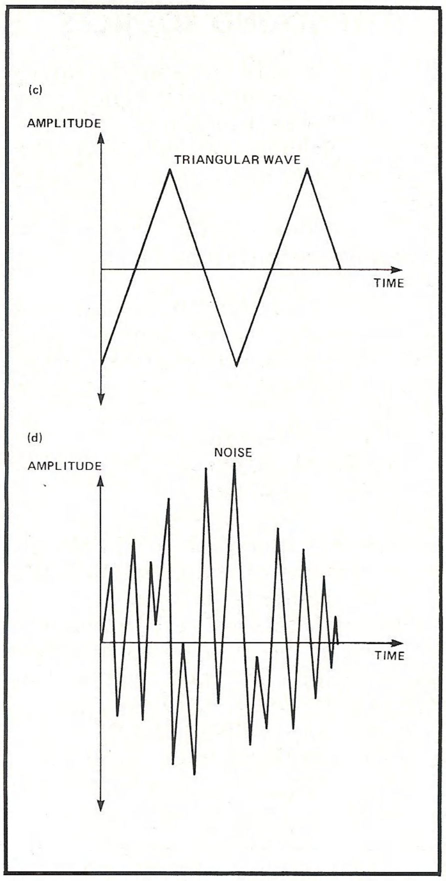
*NOTE — this register is incorrectly specified as 54288 in some versions of the User Manual.

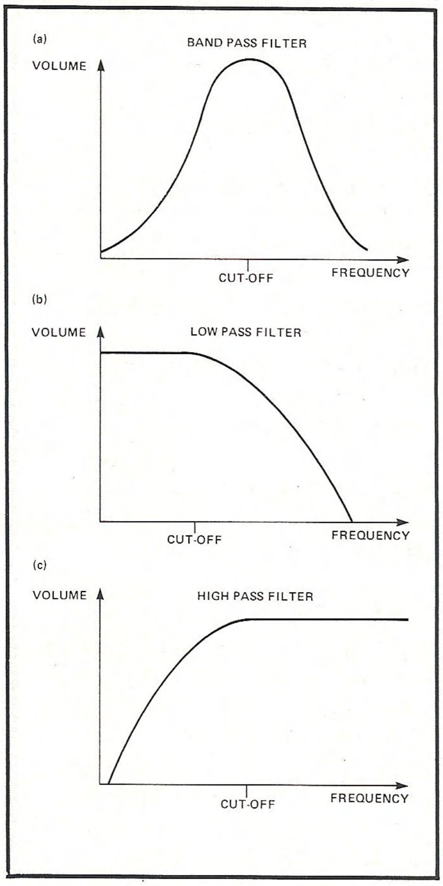
The contents of each of the control registers are very similar (we will look at the differences between them later). The bits within the registers are used as follows:

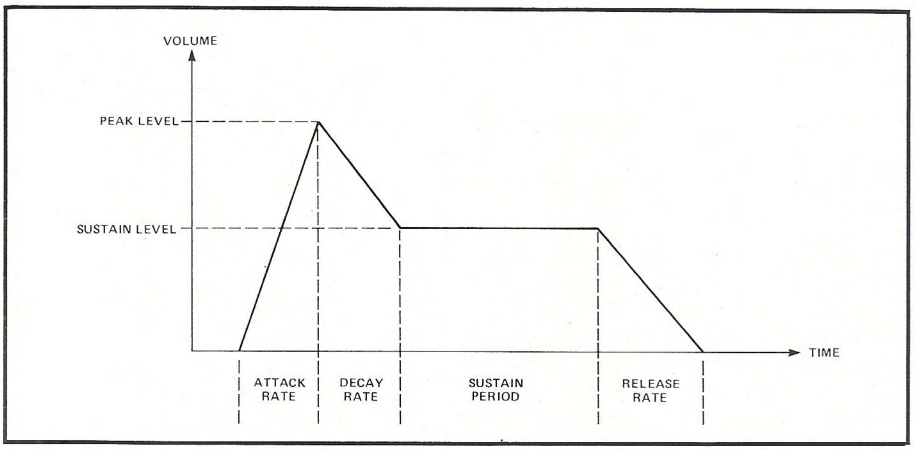
7 White noise select (1=ON)
6 Pulse waveform select (1=ON)
5 Sawtooth waveform select (1=ON)











4 Triangle waveform select (1 = ON)
3-1 do not concern us here. Leave them at 0.

0 Gate bit (1 = Start attack/decay/sustain cycle; 0 = Start release cycle)

From this you can see that there are two decimal values for each waveform type: one for the attack/decay/sustain cycle and one for the release cycle. Table 1 shows the two values of each waveform.

The procedure for playing any given note is:

- Start attack/decay/sustain cycle (POKE voice control register, Attack/decay/sustain value).
- Hold sustain for length of note (Count note length).
- When note has finished, start release (POKE voice control register, Release value).

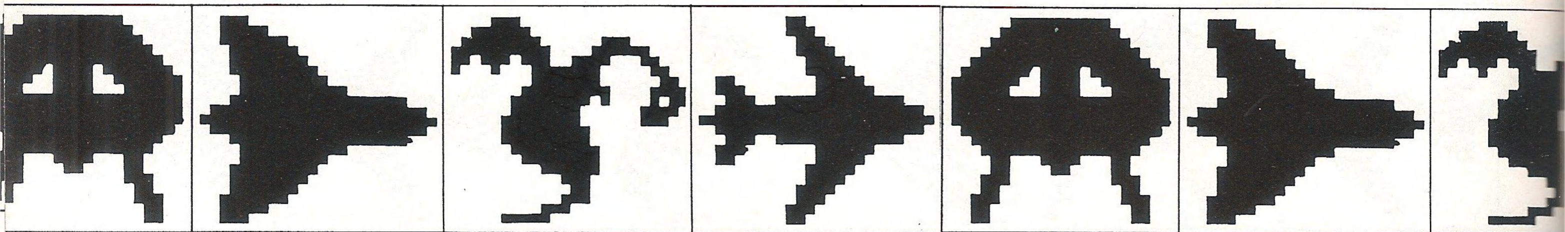
The length of the note depends on the type of note being played (crotchets, quavers and so on all have different lengths). The obvious way to time this delay is with a simple FOR-NEXT loop. You will need to determine by trial and error the delay required for a crotchet to sound the right length, but having found this value all the other note lengths are simply multiples of this value.

PITCHING IT RIGHT

Selecting the frequency of the desired note is a little bit more difficult. For each voice the frequency of the waveform generator output is determined by the values in two registers, the high frequency control register and the low frequency control register. Table 2 shows where the frequency control registers for each voice are located.

The two frequency control registers are 'read' by the SID chip as a single 16-bit register giving 65536 different selectable frequency 'steps'. The waveform generators in the SID chip can generate frequencies in the range 0 to 4000 Hz, so that each selectable frequency step changes the frequency by only 0.06 Hz. Now, we don't believe that anyone can tell the difference between two adjacent steps that are only 0.06 Hz.

Within this range of frequencies is a full eight octaves of musical notes. Appendix M in the User Manual lists the high and low frequency values needed to produce 95 of the 96 notes. (More about the 96th note later). These values should not be taken as the gospel truth — indeed a musician would throw up his hands in horror if he saw them! The reason is to do with the way in which the musical scale is constructed. The root note of the scale is usually taken to be A above middle C, which is in octave 4 of appendix M. This note has a frequency of exactly 440 Hz at concert pitch and all computers and synthesizers calculate the other notes from this one such that each note is 2^ 1/12 (the twelfth root of two) times the previous note. 'Real' notes are not exactly this distance apart but you need a good ear to be able to tell the difference. Table 3 gives a comparison between the Commodore 64 frequencies and 'real note' frequencies for A above middle C and middle C.



The result of all this is that if the values listed in appendix M do not sound right to you then feel free to change them quite drastically. (At 0.06 Hz a step some values may change by quite a lot!)

Finally, when using the rectangular waveform, there is a little bit more work to be done because we also have to set up the pulse width ratio. This value defines how wide the high part of the rectangular wave will be. (See also Fig. 8) Different values of pulse width can produce an enormous difference in the sound of rectangular waves.

For each voice the pulse width is specified as a 12-bit number, giving 4096 different steps. The high four bits of this number are specified in the High Pulse Width register and the low eight bits are specified in the Low Pulse Width register. Table 4 shows the locations of the high and low pulse width registers for each voice.

The system has been arranged so that a pulse width of 4095 gives an output that is permanently high (maximum width) and a pulse width of 0 is permanently low (minimum width). In both cases the output will be zero. Normally, of course, we will be using values in between these two — typically 2048 which produces a pure square wave.

SCALING THINGS UP

When it comes to actually storing the different notes you clearly don't want to have to store the values for them all, so some means of calculating them is required. The simplest way is to make use of the fact that the notes in one octave are twice the frequency of the notes in the octave below. This means that if we store the high and low frequency values for the highest octave we can calculate all the other octaves by successively dividing by 2.

To do this easily the high and low frequency values must be combined into one 16-bit value. This can be done as follows:

16-bit freq= (high freq)*256+low freq

Incidentally, the inverse of this which restores the high and low values from a 16-bit value is:

high freq=INT(16-bit freq/256)

low freq=16-bit freq — high freq

If you were to calculate the 16-bit value of the highest octave notes in appendix M of the User Manual you would obtain the values given in Table 5.

The procedure for calculating any note is:

- Get the name and octave of the required note.
- Read the 16-bit frequency value for the given note from the highest octave values given in Table 4.
- IF the octave number is 7

```
THEN
  Do nothing
ELSE
  FOR count=6 TO octave number STEP-1
    Frequency value =frequency value/2
  NEXT count
```

- Recover high and low frequency values from the 16-bit frequency value remaining and POKE these values into the high and low frequency control registers.

Table 1. Control register values for the various types of waveform

Waveform type	Attack/decay/sustain Gate bit=1	Release Gate bit=0
Noise	*129*	*128*
Pulse	*65*	*64*
Sawtooth	*33*	*32*
Triangle	*17*	*16*

Table 2. Locations in memory of the frequency control registers

Voice number	High Pulse Width register	Low Pulse Width register
1	*54275*	*54274*
2	*54282*	*54281*
3	*54289*	*54288*

Table 3. Frequency comparison table

Note name	Commodore frequency	Real note frequency
A above middle C	*440.0 Hz*	*440.0 Hz*
middle C	*261.6 Hz*	*264.0 Hz*

SHAPING UP

So far we have obtained a particular note being 'played' in a particular waveform. To make the note sound as though it were coming from, say, a violin or a guitar, we need to adjust its attack, decay and release rates and its sustain level. The effect that each of these parameters has on a note can be seen by referring to Fig. 9.

The Attack rate: This is the rate at which the note rises from zero to peak volume when the gate bit is set to 1.

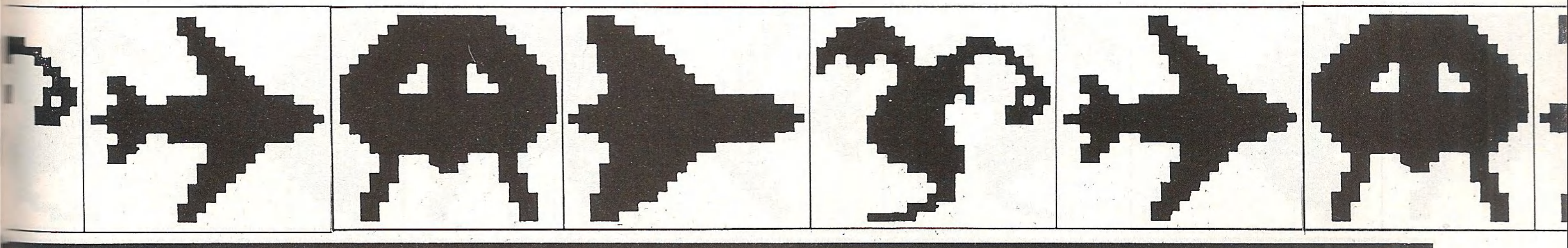
The Decay rate: This is the rate at which the note falls from its peak volume to its 'average' or sustain level.

The Sustain level: This is the proportion of the peak volume that the decay rate will fall to.

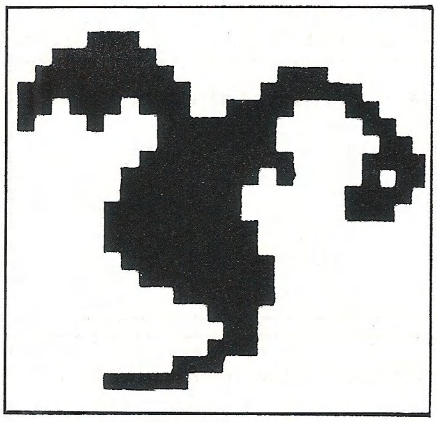
The Release rate: This is the rate at which the note dies away when the gate bit is set to 0, although very often a new note starts before the old note has fully died away.

Each of these parameters can be varied in 16 different steps and since 16 states can be coded into four bits of binary, two parameters can be fitted into one eight-bit register. For each voice, the attack and decay parameters are contained in the Attack/Decay cycle control register and the sustain and release parameters are contained in the Sustain/Release cycle control register. Table 6 shows where the two cycle control registers for each voice are located.

Within the Attack/Decay cycle control registers, the high four bits ▷



define the attack rate and the low four bits define the decay rate. Table 7 shows the rates which the different values of attack and decay can select. Within the Sustain/Release cycle control registers the high four bits define the sustain level and the low four bits define the release rate Table 7 also shows the rates which the different release values can select, and the levels which the different sustain values can select.



Appendix P of the User's Manual gives some sample ADSR settings for 'simulating' some common musical instruments. Personally we're not very impressed by the results, because there's a lot more 'real' sound than just the waveform and envelope shape. We think that it is much more satisfying to create new and different sounds. (After all you could have bought a real instrument for the price you paid for the Commodore 64!).

USING THE FILTERS

You don't have to use the filters at all if you don't need them. In fact, you can produce hours of perfectly acceptable music without ever even thinking about them. The filters are really for 'fine tuning' a sound, to get just the quality you desire. For this reason they probably have more uses in sound effects than in music.

In any case, the sort of filtering you want to do will depend on the waveform you are using. This is because the different waveforms all have different harmonic structures.

Harmonics are waves produced in addition to the generated or 'fundamental' wave. They are always integer multiples of the fundamental frequency and they have names which reflect this. For example, the second harmonic has a frequency of twice the fundamental, the third harmonic a frequency of three times the fundamental, and so on.

Some waveforms, like the sawtooth waveform, contain all the harmonics, others, like the triangular waveform, contain only the odd harmonics. In addition, the amount of each harmonic present depends upon the waveform. The triangular waveform, for example, contains harmonics in proportion to the reciprocal of the square of the har-

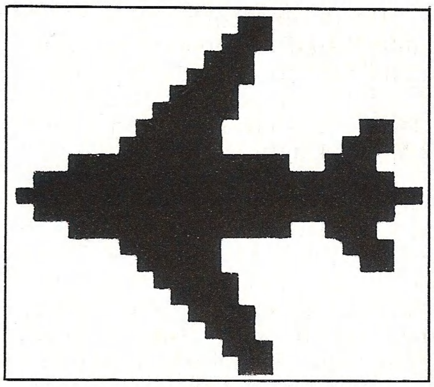


Table 4. Locations in memory of the high and low pulse width registers

Voice number	High frequency control register	Low frequency control register
1	*54273*	*54272*
2	*54280*	*54279*
3	*54287*	*54286*

Table 5. Notes in the highest octave and their 16-bit frequencies

Note name	High frequency	Low frequency	16-bit frequency
C	*137*	*43*	*35115*
C#	*145*	*83*	*37203*
D	*153*	*247*	*39415*
D#	*163*	*31*	*41759*
E	*172*	*210*	*44242*
F	*183*	*25*	*46873*
F#	*193*	*252*	*49660*
G	*205*	*133*	*52613*
G#	*217*	*189*	*55741*
A	*230*	*176*	*59056*
A#	*244*	*103*	*62567*
B	*258*	*241*	*66289**

**NOTE — this value is not included in the list in appendix M because it is greater than 65535 and therefore it cannot be 'played' by the SID chip. We have included it for completeness, partly because it is the 96th note and completes the eight octaves, but mainly so that the 'B' note of the lower octaves can be calculated from it.*

Table 6. Locations in memory of the cycle control registers.

Voice number	Attack/Decay cycle control register	Sustain/Release cycle control register
1	*54277*	*54278*
2	*54284*	*54285**
3	*54291*	*54292*

** NOTE — this register is incorrecly specified at 54286 in our version of the User Manual.*

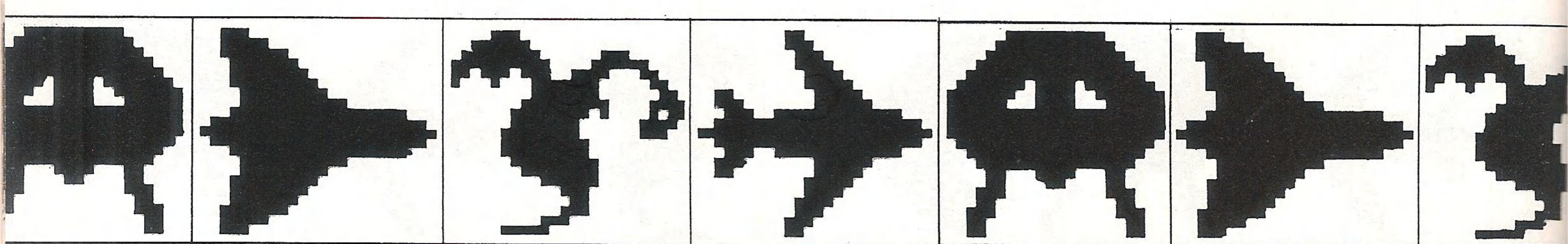


Table 7. Control register values and their effects.

Register value	Attack rate	Decay/Release rate	Sustain level
0	*2mS*	*6mS*	*0%*
1	*8mS*	*24mS*	*6%*
2	*16mS*	*48mS*	*13%*
3	*24mS*	*72mS*	*20%*
4	*38mS*	*114mS*	*26%*
5	*56mS*	*168mS*	*33%*
6	*68mS*	*204mS*	*40%*
7	*80mS*	*240mS*	*46%*
8	*100mS*	*300mS*	*53%*
9	*250mS*	*750mS*	*60%*
10	*500mS*	*1.5S*	*66%*
11	*800mS*	*2.4S*	*73%*
12	*1S*	*3S*	*80%*
13	*3S*	*9S*	*86%*
14	*5S*	*15S*	*93%*
15	*8S*	*24S*	*100%*

Table 8. Bit functions in the filter voice input control register (located at 54295 decimal).

Bit 0 — Voice 1 filter control (1 = Filter, 0 = Don't filter)

Bit 1 — Voice 2 filter control (1 = Filter, 0 = Don't filter)

Bit 2 — Voice 3 filter control (1 = Filter, 0 = Don't filter)

Bits 3 to 7 — Do not concern us here.

Table 9. Bit functions in the filter mode control register (located at 54296 decimal).

Bits 0 to 3 — Set master volume (0 = Off, 15 = Full volume)

Bit 4 — Select low pass filter (1 = On)

Bit 5 — Select band pass filter (1 = On)

Bit 6 — Select high pass filter (1 = On)

Bit 7 — Does not concern us here

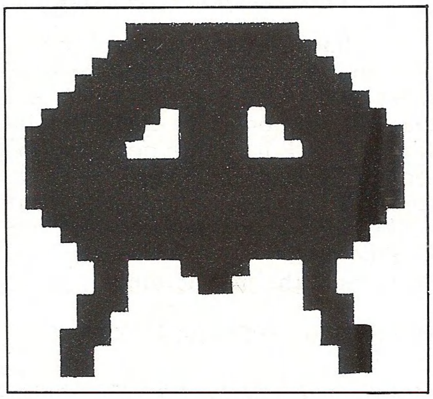
monic number, others, like the sawtooth waveform, contain harmonics in proportion to the reciprocal of the harmonic number alone.

When you decide to use the filters it is mainly the harmonics which you will be filtering out because, like the envelope shape, they contain a lot of the 'individuality' of a given sound.

The three filters are actually derived from one programmable filter within the SID chip. This filter can be programmed to act on any combination of the three voices using any combination of the three filter types (high pass, low pass and band pass). The programming is done by using two filter control registers. One, called the Voice Input control register, at location 54295 decimal selects the voices to be filtered. Table 8 shows which bits select which voices.

The other control register, the Filter Mode control register, at location 54296 decimal, is also the master volume control. Table 9 shows the utilisation of bits within this register.

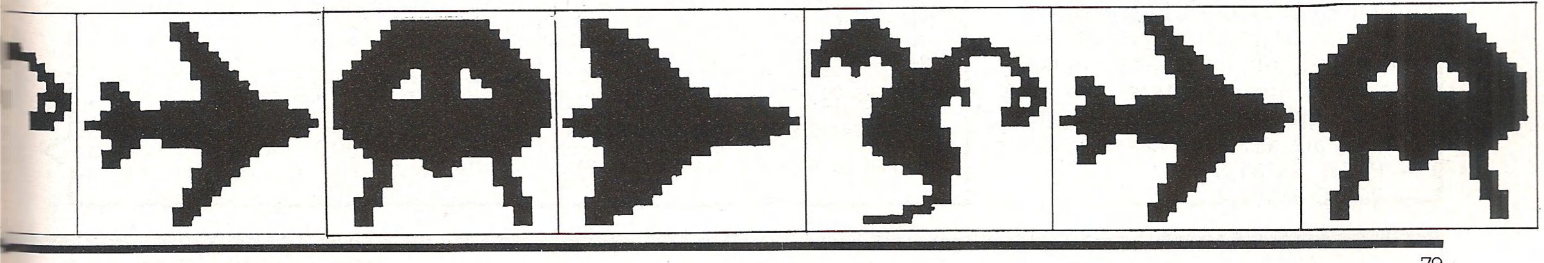
Having selected the voice and filter combinations we require, all that remains is to set the cut-off frequency for the filter. The cut-off frequency is the frequency at which the filter operates. For example, at the cut-off frequency the low pass filter will start to reject frequencies. As the frequencies increase away from the cut-off the rejection increases. (See also Fig. 10).



The cut-off frequency is specified as an 11-bit number using two registers in a rather strange way. The high eight bits of the cut-off frequency are specified in the high cut-off frequency register at location 54294 decimal. The low three bits are specified in bits 0-2 of the low cut-off frequency register at location 54293 decimal. Bits 3-7 are not used. This gives a total of 2048 different cut-off frequency steps. The SID chip filter can operate over the range 30 to 12000 Hz so that each cut-off frequency step changes the cut-off frequency by about 6 Hz. this should be fine enough control for most applications.

The procedure for using the filters is:

- Select the filter type and master volume to be used (POKE 54296, filter type and volume).
- Select the cut-off frequency (POKE 54294, high cut-off: POKE 54293, low cut-off).
- Select the voices to be filtered (POKE 542295, voice combination). ☐



MIKE ROBERTS

64 Special

Add essential features to your 64 the easy way.

The CBM64 is, as we all know, an excellent machine. Unfortunately several very useful features were omitted from either the design or the manual — extra modes, Hi-res from BASIC and accessible function keys for a start! Despair no longer, here are four simple routines that will enhance your machine and liven up your programming.

1. THE HIDDEN MODE

The 64 has four text modes of operation and, according to the manual, there is only one that the BASIC and OS will support. Not so: with one simple POKE it can be put into "Extended Background Mode."

This allows four different background colours instead of the original one.
To enter it the instruction is:

POKE 53265, (PEEK(53265)OR2 6)

And to return to normal text mode:

POKE 53265,
PEEK(53265)AND(255-(2 66))

There is one small snag though: only 64 characters may be displayed and these are the ones with POKE codes of 0 to 63 (see pages 132-3 of the User Manual). This is because the normal character set is based on eight bits, so there are 256 different characters. So the new character set has six bits for the character and two for the background colour. Thus only 64 characters and four different colours, which can be any of the 16 normally available in the ordinary mode. Foreground colours are not affected.

The new character set is composed of four blocks of the same 64 characters and each block has a different background colour. So to print an A with background colour 0, you would just PRINT a normal A; for A in background colour 1, PRINT shift A; background colour 2, a reverse field A; colour 3, a shifted reverse field A. And so on for all of the first 64 characters in the set.

The colour of each of the background colours is set by POKEing one of four registers, one for each background colour. The POKE locations are as follows:
0 — POKE 53281,X
1 — POKE 53282,X

```
100 POKE53265,PEEK(53265)OR2↑6:REM SET BIT 6 OF VIC MODE REGISTER
110 REM AS THIS ENABLES EXTENDED BACKGROUND MODE
120 REM THE NEXT SECTION SETS THE 4 DIFFERANT BACKGROUND COLOURS
130 POKE53281,0:REM SET BACKGROUND COLOUR 0 FOR CHARACTERS 0 TO 63
140 POKE53282,1:REM SET BACKGROUND COLOUR 1 FOR CHARACTERS 64 TO 127
150 POKE53283,2:REM SET BACKGROUND COLOUR 2 FOR CHARACTERS 128 TO 191
160 POKE53284,3:REM SET BACKGROUND COLOUR 3 FOR CHARACTERS 192 TO 255
170 REM THE NEXT SECTION WILL DISPLAY THE 64 DISPLAYABLE CHARACTERS
180 REM IN THEIR FOUR DIFFERANT BACKGROUND COLOURS AND 16 FORGROUND COLOURS
190 PRINTCHR$(147):FORI=0TO999:REM CLEAR SCREEN AND START LOOP
200 POKE1024+I,IAND255:REM POKE CHARACTERS
210 POKE55296+I,IAND15:REM POKE FORGROUND COLOURS
220 NEXT:REM COMPLETE LOOP
230 POKE53281,(PEEK(53281)AND15)+1:REM INCREMENT THE BACKGROUND COLOURS
240 POKE53282,(PEEK(53282)AND15)+1
250 POKE53283,(PEEK(53283)AND15)+1
260 POKE53284,(PEEK(53284)AND15)+1
270 FORI=0TO100:NEXT:REM DELAY LOOP
280 GOTO230:REM LOOP ROUND FOR INFINITY
```

```
10 REM*********************************
20 REM**                             **
30 REM** HI RESOLUTION GRAPHICS MODE **
40 REM**                             **
50 REM**      ON THE COMMODORE 64    **
60 REM**                             **
70 REM** BY MIKE ROBERTS AUGUST 1983 **
80 REM**                             **
90 REM*********************************
98 :
99 REM SET HIRES MODE
100 POKE56576,148:POKE53272,24:POKE53265,59:SC=57344 CL=50176:RETURN
198 :
199 REM SET TEXT MODE
200 POKE56576,151:POKE53272,21:POKE53265,155:RETURN
298 :
299 REM CLEAR GRAPHICS SCREEN
300 BC=PEEK(53281):QQ=1000+CL:FORI=CLTOQQ:POKEI,BC:NEXT:REM CLEAR GRAPHICS
310 QQ=SC+8000:FORI=SCTOQQ:POKEI,0:NEXT:REM CLEAR COLOUR
320 RETURN
398 :
399 REM TRANSFER TEXT SCREEN TO GRAPHICS SCREEN
400 BC=(PEEK(53281)AND15):QQ=1000:REM FIND BACKGROUND COLOUR
410 FORI=0TOQQ:POKEI+CL,BC+(PEEK(55296+I)AND15)*16:NEXT:REM FORGROUND COLOUR
420 POKE56333,127:POKE1,51:REM PAGE OUT I/O AND PAGE IN CHARACTER GENERATOR
430 QQ=SC:FORI=0TO1000:TC=PEEK(I+1024):REM FIND CHARACTER CODE ON SCREEN
440 GC=53248+TC*8+((CS=1)*-2048):FORJ=0TO7:POKEQQ+J,PEEK(GC+J):NEXTJ:QQ=QQ+8
445 REM FIND AND PLOT BIT IMAGE OF CHARACTER
450 NEXTI:POKE1,55:POKE56333,129:RETURN:REM PAGE IN I/O AND OUT CHAR GENERATOR
498 :
499 REM PLOT X,Y : FORGROUND COLOUR IN FC : BACKGROUND COLOUR IN BC
500 IFX<0ORX>319ORY<0ORY>199THENGOSUB200:PRINT"PLOT ERROR":STOP
505 REM CHECK THAT COORDINATES ARE WITHIN BOUNDS
510 XB=INT(X/8):X=X-XB*8:REM FIND X BLOCK
520 YB=INT(Y/8):Y=Y-YB*8:REM FIND Y BLOCK
530 POKECL+XB+YB*40,FC*16+BC:REM PLOT COLOUR OF DOT IN CELL
540 QQ=SC+Y+YB*320+XB*8:POKE253,INT(QQ/256):POKE252,QQ-(INT(QQ/256)*256)
550 POKE255,2↑(7-X):POKE254,RW:SYS49152:RETURN:REM GET DOT LOCATION ON GRAPHICS
560 REM SCREEN, SET UP AND CALL MACHINE CODE MEM PAGEING AND PLOTTING ROUTINE
598 :
599 REM PRINT STRING IN X$ ONTO GRAPHICS SCREEN
600 QQ%=LEN(X$):IFQQ%<1ORX>39ORX<0ORY<0ORY>24THENGOSUB200:PRINT"PRINT ERROR":ST
OP
605 REM CHECK THAT COORDINATES ARE WITHIN BOUNDS
610 QQ%=0:FORI=1TOLEN(X$):QQ=ASC(MID$(X$,I,1)):IFQQAND96THENQQ%=QQ%+1
615 REM FIND NUMBER OF PRINTABLE CHARACTERS IN THE STRING
620 NEXTI:QQ%=QQ%-1:QQ=SC+Y*320+X*8:XX=CL+Y*40+X
625 REM FIND APPROPRIATE SCREEN MEMORY LOCATIONS
630 IFY*40+X+QQ%>1000THENGOSUB200:PRINT"LENGTH ERROR":STOP
635 REM CHECK THAT LENGTH OF STRING WILL FIT ON THE SCREEN
640 FORI=XXTOQQ%+XX:POKEI,FC*16+BC:IFHGTHENPOKEI+40,FC*16+BC
642 REM PERFORM COLOUR CHANGING
645 NEXTI:PRINT"⊐"X$:POKE56333,127:POKE1,51:REM PAGE OUT I/O AND CHAR GEN IN
650 FORI=0TOQQ%:TC=PEEK(I+1024):GC=53248+((CS=1)*-2048)+TC*8:IFHG=0THEN680
655 REM GET CHARACTER VALUES
660 FORJ=0TO7STEP2:POKEQQ+J,PEEK(GC):POKEQQ+J+1,PEEK(GC):GC=GC+1:NEXTJ:QQ=QQ+32
0
665 REM DO DOUBLE HEIGHT CHARACTERS
```

2 — POKE 53283,X
3 — POKE 53284,X
X would be any number between 0 and 15 to define a colour in the normal way.

The program demonstrates the use of this technique and will cycle through all the colours and background colours. Overall the advantage of this mode is its similarity to the normal start up mode. Programs can still be listed, edited and written but this mode comes into operation when something is being printed on the screen under program control.

2. HI-RES GRAPHICS

You will be aware that the 64 allows high resolution graphics — after all, they appear on many commercial games. But the trouble is that the BASIC supplied with the machine does not support this feature. In fact the 64 can have six different modes of screen operation, five of which the OS does not support. So you can't write programs in BASIC that have hi-res graphics.

The following routines will help in the writing of programs using the highest resolution graphics modes — 320 by 200 dots and 16 colours on screen at once. By using a page switching method we can have a graphics screen that takes up no user memory at all as it occupies the same memory space as BASIC.

Hi-res functions are called by subroutines that must be GOSUBed first via BASIC. Before a routine is called, however, some variables may need to be set. In fact many variables are used by these routines and it is wise to avoid these as they may corrupt your program or crash the system.

3. FUNCTION KEYS

The potentially useful function keys to the right of the keyboard are virtually inaccessible for any practical purpose. My solution to this was to think about how the BBC micro has useful function keys and come up with this program which offers you 16 user-defined function keys.

This is done as follows:
f1, f3, f5, f7 become the four normal function keys;
f2, f4, f6, f8 become shifted f1 to f7;
f9, f11, f13, f15 become f1 to f7 in conjunction with the Commodore key;
f10, f12, f14, f16 become f1 to f7 used with the CTRL key.

This program takes up no user memory for BASIC, because it sits in memory locations C000 to D000, but check these before using this with another program in case it uses

```
670 FORJ=0TO7STEP2:POKEQQ+J,PEEK(GC):POKEQQ+J+1,PEEK(GC):GC=GC+1:NEXTJ
675 IFUL%THENPOKEQQ+7,255-PEEK(GC-1):REM DO UNDERLINING
677 QQ=QQ-312:GOTO690
680 FORJ=0TO7:POKEQQ+J,PEEK(GC+J):NEXTJ:QQ=QQ+8:IFUL%THENPOKEQQ-1,255-PEEK(GC)
685 REM DO SINGLE HEIGHT CHARACTERS AND UNDERLINING IF NECESSARY
690 NEXTI:POKE1,55:POKE56333,129:RETURN:REM PAGE I/O BACK IN
698 :
699 REM DATA FOR MACHINE CODE PAGE SWITCHING AND PLOTTING
700 DATA 173,13,220,72,169,0,141,13,220,165,1,41,253,133,1,160,0,177
710 DATA 252,166,254,224,0,240,254,5,255,145,252,165,1,9,2,133,1,104,141,13,220
720 DATA 96,72,165,255,73,255,133,255,104
730 I=49152:QQ=0:DATA 37,173,13,220,72,169,0,-1
740 READQQ%:IFQQ%=-1THEN760:REM READ AND
750 POKEI,QQ%:I=I+1:QQ=QQ+QQ%:GOTO740:REM POKE IN DATA
760 IFQQ<>6880THENPRINT"ERROR IN DATA LINES 700 TO 730":STOP:REM CHECK DATA IS
770 RETURN:REM OK, IF SO THEN RETURN
 1000 GOSUB700:GOSUB100:X=312:Y=192:RW=1:FC=1:BC=0:GOSUB500
READY.
```

* * *

HOW IT RUNS

110	*set extended background colour mode*
130	*set background colour 0*
140	*set background colour 1*
150	*set background colour 2*
160	*set background colour 3*
190-220	*display character set on screen in 16 colours*
230-260	*increment the four colour registers*
280	*delay loop*
290	*loop to cycle through colours*

```
10070 DIM FK$(16)
10080 FK$(1)="RUN"+CHR$(13):REM  F1 & RETURN
10090 FK$(2)="LIST":REM          F3
10100 FK$(3)="CONT"+CHR$(13):REM F5 & RETURN
10110 FK$(4)="LIST"+CHR$(13):REM F7 & RETURN
10120 FK$(5)="POKE":REM          F2
10130 FK$(6)="PEEK(":REM         F4
10140 FK$(7)="FOR":REM           F6
10150 FK$(8)="NEXT":REM          F8
10160 FK$(9)="LEFT$(":REM        F9
10170 FK$(10)="RIGHT$(":REM      F11
10180 FK$(11)="MID$(":REM        F13
10190 FK$(12)="CHR$(":REM        F15
10200 FK$(13)="POKE650,":REM     F10 REPEAT CONTROL
10210 FK$(14)="POKE53280,":REM   F12 BORDER COLOUR
10220 FK$(15)="POKE53281,":REM   F14 BACKGROUND COLOUR
10230 FK$(16)="53248":REM        F16 LOCATION OF START OF VIC2 CHIP
10240 SL=49152
10250 READB:IFB=999THEN10270
10260 POKESL,B:SL=SL+1:GOTO10250
10270 FORD=1TO16:F$=FK$(D)
10280 FORC=1TOLEN(F$):POKESL,ASC(MID$(F$,C,1)):SL=SL+1:NEXTC:POKESL,0
10290 SL=SL+1:NEXTD
10300 SYS49152:POKE251,SL-(INT(SL/256)*256):POKE252,INT(SL/256)
10310 PRINT:INPUT"DO YOU WANT TO SAVE M/C VERSION";A$:IFLEFT$(A$,1)="N"THENEND
10315 POKE253,1:INPUT"DISK OR TAPE ( D/T )";DK$:IFLEFT$(DK$,1)="D"THENPOKE253,8
10320 POKE814,PEEK(45):POKE815,PEEK(46)
10325 POKE43,0:POKE44,192:POKE45,PEEK(251):POKE46,PEEK(252)
10340 SAVE"M/C USRKEYS",PEEK(253)
10350 POKE43,1:POKE44,8:POKE45,PEEK(814):POKE46,PEEK(815)
10360 END
10370 DATA160,0,185,142,193,32,209,192,200,192,29,208,245,173,13,220,72
10380 DATA169,0,141,13,220,162,94,142,20,3,162,192,142,21,3,162,92,142,17
10390 DATA3,162,193,142,18,3,162,0,142,19,3,162,1,134,45,162,8,134,46
10400 DATA160,0,177,45,230,45,208,2,230,46,201,0,208,244,177,45,208,242
10410 DATA230,45,208,2,230,46,177,45,208,232,230,45,208,2,230,45,104,141
10420 DATA13,220,96,165,197,201,3,48,122,201,7,16,118,174,19,3,224,0
10430 DATA208,108,166,157,224,0,240,102,162,255,142,19,3,201,3,208,2,169
10440 DATA7,56,233,4,78,141,2,144,6,24,105,4,76,161,192,78,141,2,144,6
10450 DATA24,105,8,76,161,192,78,141,2,144,3,24,105,12,162,201,134,251
10460 DATA162,193,134,252,160,0,170,177,251,201,0,208,5,224,0,240,10,202
10470 DATA230,251,208,2,230,252,76,172,192,200,177,251,201,0,240,20,201
10480 DATA13,240,9,32,209,192,76,192,192,108,38,3,162,1,134,198,141,119
10490 DATA2,76,49,234,162,0,142,19,3,173,141,2,41,4,240,241,162,0,189
10500 DATA12,193,197,197,240,8,201,0,240,228,232,76,236,192,138,10,170,189
10510 DATA19,193,133,251,232,189,19,193,133,252,108,251,0,46,21,9,20,10
10520 DATA18,0,31,193,54,193,61,193,68,193,76,193,84,193,173,0,220,72
10530 DATA169,2,141,0,220,173,1,220,201,187,208,249,104,141,0,220,76,49
10540 DATA234,162,0,134,212,76,49,234,162,0,134,216,76,49,234,162,0,142
10550 DATA138,2,76,49,234,162,255,142,138,2,76,49,234,162,96,142,138,2
10560 DATA76,49,234,32,139,193,192,0,208,3,76,0,192,173,13,220,72,169,0
10570 DATA141,13,220,162,49,142,20,3,162,234,142,21,3,104,141,13,220,160
10580 DATA0,185,171,193,32,209,192,200,192,29,208,245,96,108,3,0,85,83
10590 DATA69,82,32,68,69,70,73,78,65,66,76,69,32,75,69,89,83,32,13,69
10600 DATA78,65,66,76,69,68,0,85,83,69,82,32,68,69,70,73,78,65,66,76
10610 DATA69,32,75,69,89,83,32,13,68,73,83,65,66,76,69,68,0,0,999

READY.
```

▷

these locations for machine code. Strings of characters up to 200 characters long can be used.

As well as the function keys, the program adds a number of control functions which are accessed by pressing CRTL in conjunction with another key — see figure 1. I have also given a list of the normal CTRL for the 64 as these may be useful for you — see figure 2.

When the BASIC loader program has been run, you are given the option of saving the program in machine code format on tape. This means that it will load faster and, unlike the BASIC loader, can be loaded on top pf a BASIC program without destroying it. Load the machine code with LOAD"",1,1 and then initialise with SYS 49152.

The system can be returned to normal at anytime by executing PRINT USR(1) and re-enabled by executing PRINT USR(0).

4. MULTI-COLOURED FLASHING

If you fancy the sort of flashing characters you see on Prestel or Ceefax then this last program will provide them for you.

It uses the multi-coloured text mode, which is more complex than the other three text modes as the whole character set needs to be redefined. However, the advantages of this mode do outweigh the disadvantages in certain applications. Each dot of the character matrix can be one of four colours: background, foreground and two others defined elsewhere.

These two other colours have their definitions in two of the 6569 VIC chip's registers and if these registers are changed then the effect is felt all over the screen and any character using them will have its colour changed. A list of these registers and their locations is given below.

So the principle behind this program is that half of the character set (the reverse field half) is defined using the special colours (one foreground, one background) and the other half is as normal, except that the characters are only four dots wide.

To change the registers at regular intervals we redirect the interrupt routine every 1/60th of a second. To turn on the flash routine use instruction SYS 49152 and SYS 49165 to turn it off again.

There are four modes of operation of the flashing controlled by POKEing the flash control register: 0, no flashing of foreground or background; 1, flashing background

VARIABLES USED

SC	holds the memory location of the start of the hi-res screen in memory
CI	holds the start address of the colour map in memory

QQ,QQ%,I,J,TC,GC,XB,YB,XX, are all temporary variables used in the routines

HOW IT RUNS

100	*sets graphics screen into operation mode*
200	*sets text mode (usual 64 operation mode)*
300-399	*clears graphics screen and sets background colour to that of normal text mode (memory location 53281)*
400-499	*copies text screen to graphics screen (CS set to 1 produces lower case copy, CS sets to 0 upper case)*
500-599	*plots point on graphics screen at specified coordinates. The following variables need to be set on entry to this routine:* *FC foreground colour (0-15)* *BC background colour (0-15)* *RW rub out (1) or write (0) dot to screen* *X position of start of text (0-319; origin is top left)* *Y position of start of text (0-199)*
600-699	*prints text string X$ to graphics screen. Again the following variables need to be set:* *X,Y as above* *FC,BC as above* *CS CaSe change (1 for lower, 0 for upper)* *HG character height (1 for double, 0 for normal)* *UL% underlines text if set to 1*
700-799	*machine code for page switch and plot must be run before use of the plot routine, starts at location 49152 but is relocatable*

CTRL FUNCTION TABLE

Key Pressed (with control)	*Function*
@	*will totally stop system, to return to normal press CTRL AND↑, or the usual stop/restore*
A	*makes all keys repeat*
D	*disables repeat key*
C	*only cursor control keys repeat, as normal*
F	*escape from quote mode*
W	*escape from insert mode*

'NORMAL' CTRL FUNCTIONS

Key Pressed (with control)	*Function*
E	*same as WHT or CTRL-2*
H	*disables lower/upper case shift*
I	*enables lower/upper case shift*
M	*same as return key*
N	*switch to lower case*
Q	*same as cursor down*
R	*same as rvs on*
S	*same as home*
T	*same as delete*
£	*same as RED or CTRL-3*
↑	*same as GRN or CTRL-6*

only; 2, flashing foreground only; 3, flashing background and foreground. So POKE 822,1 would give mode 1 as the location is 822; to change the speed register at location 823, POKE 823,x where x is the speed of flashing in 1/60th of a second.

The colours that alternate for the foreground are in locations 821 (next colour) and 53283 (current colour); thus POKE 821,1 and POKE 53283,0 would make the foreground flash between black and white.

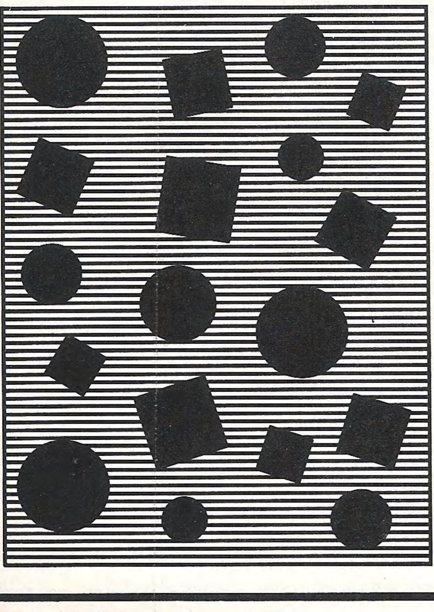
The colours that alternate for the background are in locations 820 (next colour) and 53282 (current colour); thus POKE 820,2 and POKE 53282,3 would make the background flash between red and cyan.

Unfortunately only eight colours can be used in this mode and, for this application, they must have values of 8-15. So to get black text use the Commodore key in conjunction with key 1. Using values less than eight will merely produce on-screen garbage.

So to print flashing text you must first set up the above POKEs and enable the system. Remember that the character printing colour must be greater than eight, although only the lower eight colours are displayed, and the text must be printed in reverse field.

Of course, normal procedures can be carried out for non-flashing characters, except that reverse field cannot be accessed. To return to normal mode POKE 53272,21:POKE 53270,200; to get back to flashing mode use POKE 53272,19:POKE 53270,216.

Note that before you type in or load the program, you must execute POKE 44,24:POKE 6144,0:NEW in order to reserve memory space for the new character set. This will destroy any program currently in memory.

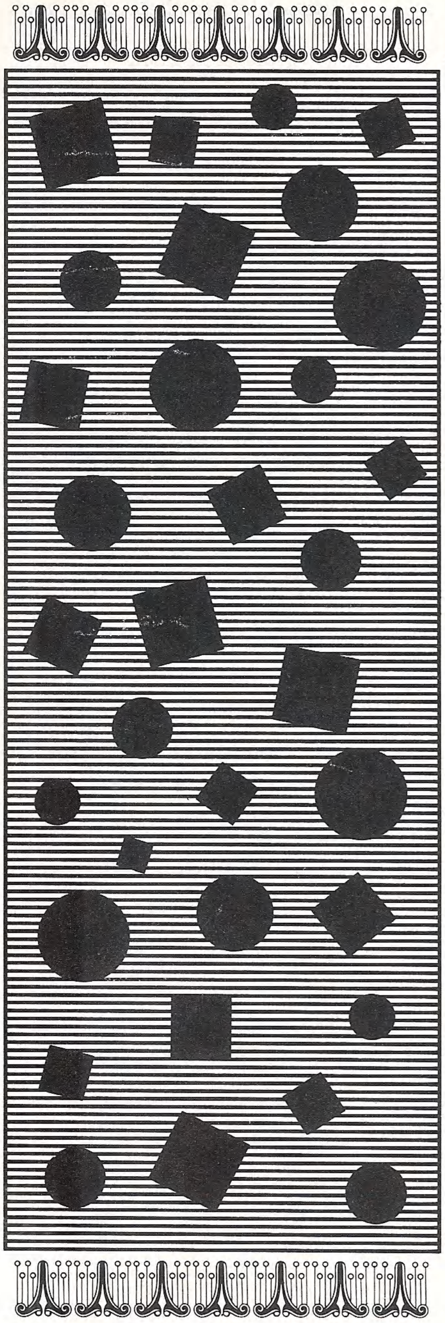


REGISTERS USED

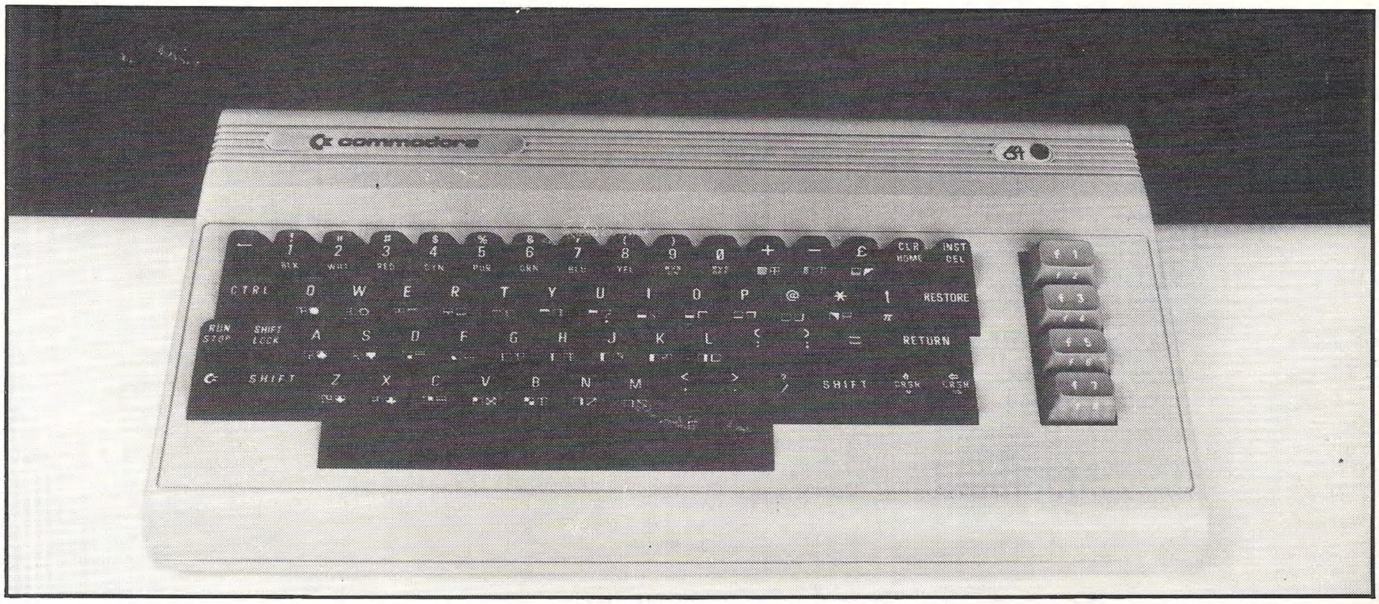
REGISTERS USED	**Location**	**Effect**
Background colour 1	*820*	*Next background colour to be used*
Background colour 2	*53282*	*Current background colour of flashing character*
Foreground colour 1	*821*	*Next foreground colour*
Foreground colour 2	*53283*	*Current foreground colour*
Control register	*822*	*Mode of flashing operation*
Time register	*823*	*Delay between flashes*
VIC mode register 1	*53270*	*Enables multi-colour mode*
VIC mode register 2	*53272*	*Relocates character generator*

Enable flash: SYS 49152
Disable flash: SYS 49165

```
100 REM DATA FOLLOWS FOR FLASHING CHARACTER SET
110 DATA 63, 51, 51, 51, 48, 46, 63, 0, 12, 63, 51, 63, 51, 51, 51, 0, 63, 51
120 DATA 51, 60, 51, 51, 63, 0, 15, 48, 48, 48, 48, 48, 15, 0, 60, 51, 51, 51
130 DATA 51, 51, 60, 0, 63, 48, 48, 60, 48, 48, 63, 0, 63, 48, 48, 60, 48, 48
140 DATA 48, 0, 63, 51, 46, 51, 51, 51, 63, 0, 51, 51, 51, 63, 51, 51, 51, 0
150 DATA 63, 12, 12, 12, 12, 12, 63, 0, 63, 12, 12, 12, 12, 60, 60, 0, 51, 51
160 DATA 60, 60, 51, 51, 51, 0, 48, 48, 48, 48, 48, 48, 63, 0, 51, 63, 63, 63
170 DATA 51, 51, 51, 0, 51, 51, 51, 63, 63, 63, 51, 0, 12, 51, 51, 51, 51, 51
180 DATA 12, 0, 60, 51, 51, 60, 48, 48, 48, 0, 63, 63, 51, 51, 63, 63, 3, 0
190 DATA 60, 51, 51, 60, 60, 51, 51, 0, 12, 51, 46, 12, 3, 51, 12, 0, 63, 12
200 DATA 12, 12, 12, 12, 12, 0, 51, 51, 51, 51, 51, 63, 12, 0, 51, 51, 51, 51
210 DATA 51, 12, 12, 0, 195, 195, 195, 195, 255, 255, 195, 0, 51, 51, 63, 12
220 DATA 63, 51, 51, 0, 51, 51, 63, 63, 12, 12, 12, 0, 63, 3, 12, 12, 12, 48
230 DATA 63, 0, 60, 48, 48, 48, 48, 48, 60, 0, 12, 51, 48, 60, 48, 51, 63, 0
240 DATA 60, 12, 12, 12, 12, 12, 60, 0, 0, 12, 63, 63, 12, 12, 12, 12, 0, 12
250 DATA 48, 255, 255, 48, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 12, 12, 12, 0, 0
260 DATA 12, 0, 51, 51, 51, 0, 0, 0, 0, 0, 51, 51, 255; 51, 255, 51, 51, 0, 12
270 DATA 63, 192, 60, 3, 252, 12, 0, 51, 51, 3, 12, 48, 51, 51, 0, 60, 195, 60
280 DATA 60, 243, 204, 51, 0, 3, 12, 48, 0, 0, 0, 0, 0/ 15, 60, 48, 48, 48, 60
290 DATA 15, 0, 60, 15, 3, 3, 3, 15, 60, 0, 0, 195, 60, 255, 60, 195, 0, 0, 0
300 DATA 12, 12, 63, 63, 12, 12, 0, 0, 0, 0, 0, 0, 12, 12, 48, 0, 0, 0, 63, 63
310 DATA 0, 0, 0, 0, 0, 0, 0, 0, 12, 12, 0, 0, 3, 12, 12, 48, 48, 192, 0, 63
320 DATA 63, 51, 51, 51, 63, 63, 0, 12, 60, 12, 12, 12, 12, 63, 0, 63, 51, 3
330 DATA 12, 48, 51, 63, 0, 12, 51, 3, 12, 3, 51, 12, 0, 3, 15, 51, 51, 63, 3
340 DATA 3, 0, 63, 48, 60, 3, 3, 51, 12, 0, 15, 48, 48, 63, 51, 51, 12, 0, 63
350 DATA 3, 3, 15, 12, 12, 12, 0, 63, 51, 51, 63, 51, 51, 63, 0, 63, 51, 51
360 DATA 63, 3, 51, 63, 0, 0, 0, 12, 12, 0, 12, 12, 0, 0, 0, 12, 12, 0, 12, 12
370 DATA 48, 3, 12, 48, 48, 48, 12, 3, 0, 0, 0, 63, 63, 0, 63, 63, 0, 48, 12
380 DATA 3, 3, 3, 12, 48, 0, 12, 51, 3, 12, 12, 0, 12, 0, 0, 0, 0, 255, 255, 0
390 DATA 0, 0, 12, 12, 63, 63, 63, 12, 63, 0, 12, 12, 12, 12, 12, 12, 12, 12
400 DATA 0, 0, 0, 255, 255, 0, 0, 0, 0, 0, 255, 255, 0, 0, 0, 0, 0, 255, 255
410 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 0, 0, 48, 48, 48, 48, 48, 48, 48
420 DATA 48, 12, 12, 12, 12, 12, 12, 12, 12, 0, 0, 0, 240, 252, 60, 12, 12, 48
430 DATA 48, 60, 63, 15, 0, 0, 0, 12, 12, 60, 60, 240, 240, 0, 0, 192, 192
440 DATA 192, 192, 192, 192, 255, 255, 192, 192, 240, 48, 60, 12, 15, 3, 3, 3
450 DATA 15, 12, 60, 48, 240, 192, 255, 255, 192, 192, 192, 192, 192, 192, 255
460 DATA 255, 3, 3, 3, 3, 3, 3, 0, 60, 255, 255, 255, 255, 60, 0, 0, 0, 0, 0
470 DATA 0, 255, 255, 0, 51, 63, 63, 63, 63, 12, 12, 0, 48, 48, 48, 48, 48, 48
480 DATA 48/ 48, 0, 0, 0, 15, 63, 60, 48, 48, 195, 195, 255, 60, 60, 255, 195
490 DATA 195, 0, 60, 255, 195, 195, 255, 60, 0, 12, 12, 51, 51, 12, 12, 63, 0
500 DATA 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 63, 63, 63, 12, 12, 0, 48, 48
510 DATA 48, 255, 255, 48, 48, 48, 192, 192, 48, 48, 192, 192, 48, 48, 48, 48
520 DATA 48, 48, 48, 48, 48, 48, 0, 0, 63, 63, 51, 51, 51, 0, 255, 255, 63, 63
530 DATA 15, 15, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 240, 240, 240, 240, 240, 240
540 DATA 240, 240, 0, 0, 0, 0, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0
550 DATA 0, 0, 0, 0, 0, 0, 255, 192, 192, 192, 192, 192, 192, 192, 192, 204
560 DATA 204, 51, 51, 204, 204, 51, 51, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0
570 DATA 204, 204, 51, 51, 255, 255, 252, 252, 240, 240, 192, 192, 3, 3, 3, 3
580 DATA 3, 3, 3, 3, 48, 48, 48, 63, 63, 48, 48, 48, 0, 0, 0, 0, 15, 15, 15
590 DATA 15, 48, 48, 48, 63, 63, 0, 0, 0, 0, 0, 0, 252, 252, 12, 12, 12, 0, 0
600 DATA 0, 0, 0, 0, 255, 255, 0, 0, 0, 63, 63, 48, 48, 48, 48, 48, 48, 255
610 DATA 255, 0, 0, 0, 0, 0, 0, 255, 255, 48, 48, 48, 48, 48, 48, 240, 240, 48
620 DATA 48, 48, 192, 192, 192, 192, 192, 192, 192, 192, 192, 192, 192, 192
630 DATA 192, 192, 192, 192, 3, 3, 3, 3, 3, 3, 3, 3, 255, 255, 0, 0, 0, 0, 0
640 DATA 0, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 3, 3
650 DATA 3, 3, 3, 3, 255, 255, 0, 0, 0, 0, 240, 240, 240, 240, 15, 15, 15, 15
660 DATA 0, 0, 0, 0, 48, 48, 48, 240, 240, 0, 0, 0, 240, 240, 240, 240, 0, 0
670 DATA 0, 0, 240, 240, 240, 240, 15, 15, 15, 15, 106, 102, 102, 102, 101
680 DATA 101, 106, 85, 89, 106, 102, 106, 102, 102, 102, 85, 106, 102, 102
690 DATA 105, 102, 102, 106, 85, 90, 101, 101, 101, 101, 101, 90, 85, 105, 102
700 DATA 102, 102, 102, 102, 105, 85, 106, 101, 101, 105, 101, 101, 106, 85
710 DATA 106, 101, 101, 105, 101, 101, 101, 85, 106, 102, 101, 102, 102, 102
720 DATA 106, 85, 102, 102, 102, 106, 102, 102, 102, 85, 106, 89, 89, 89, 89
730 DATA 89, 106, 85, 106, 89, 89, 89, 89, 105, 105, 85, 102, 102, 105, 105
740 DATA 102, 102, 102, 85, 101, 101, 101, 101, 101, 101, 106, 85, 102, 106
750 DATA 106, 106, 102, 102, 102, 85, 102, 102, 102, 106, 106, 106, 102, 85
760 DATA 89, 102, 102, 102, 102, 102, 89, 85, 105, 102, 102, 105, 101, 101
770 DATA 101, 85, 106, 106, 102, 102, 106, 106, 86, 85, 105, 102, 102, 105
780 DATA 105, 102, 102, 85, 89, 102, 101, 89, 86, 102, 89, 85, 106, 89, 89, 89
790 DATA 89, 89, 89, 85, 102, 102, 102, 102, 102, 106, 89, 85, 102, 102, 102
800 DATA 102, 102, 89, 89, 85, 150, 150, 150, 150, 170, 170, 150, 85, 102, 102
810 DATA 106, 89, 106, 102, 102, 85, 102, 102, 106, 106, 89, 89, 89, 85, 106
820 DATA 86, 89, 89, 89, 101, 106, 85, 105, 101, 101, 101, 101, 101, 105, 85
830 DATA 89, 102, 101, 105, 101, 102, 106, 85, 105, 89, 89, 89, 89, 89, 105
840 DATA 85, 85, 89, 106, 106, 89, 89, 89, 89, 85, 89, 101, 170, 170, 101, 89
```



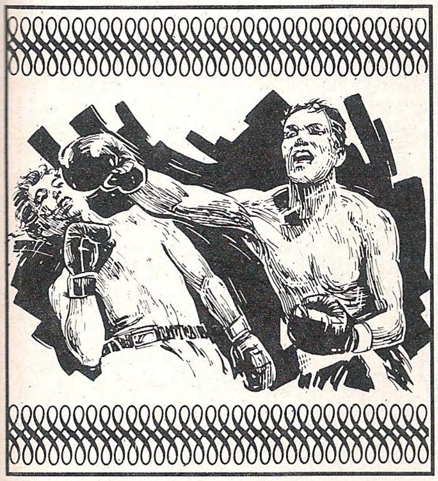
```
850 DATA 85, 85, 85, 85, 85, 85, 85, 85, 85, 89, 89, 89, 89, 85, 85, 89, 85
860 DATA 102, 102, 102, 85, 85, 85, 85, 85, 102, 102, 170, 102, 170, 102, 102
870 DATA 85, 89, 106, 149, 105, 86, 169, 89, 85, 102, 102, 86, 89, 101, 102
880 DATA 102, 85, 105, 150, 105, 105, 166, 153, 102, 85, 86, 89, 101, 85, 85
890 DATA 85, 85, 85, 90, 105, 101, 101, 101, 105, 90, 85, 105, 90, 86, 86, 86
900 DATA 90, 105, 85, 85, 150, 105, 170, 105, 150, 85, 85, 85, 89, 89, 106
910 DATA 106, 89, 89, 85, 85, 85, 85, 85, 85, 89, 89, 101, 85, 85, 85, 106
920 DATA 106, 85, 85, 85, 85, 85, 85, 85, 85, 89, 89, 85, 85, 86, 89, 89, 101
930 DATA 101, 149, 85, 106, 106, 102, 102, 102, 106, 106, 85, 89, 105, 89, 89
940 DATA 89, 89, 106, 85, 106, 102, 86, 89, 101, 102, 106, 85, 89, 102, 86, 89
950 DATA 86, 102, 89, 85, 86, 90, 102, 102, 106, 86, 86, 85, 106, 101, 105, 86
960 DATA 86, 102, 89, 85, 90, 101, 101, 106, 102, 102, 89, 85, 106, 86, 86, 90
970 DATA 89, 89, 89, 85, 106, 102, 102, 106, 102, 102, 106, 85, 106, 102, 102
980 DATA 106, 86, 102, 106, 85, 85, 85, 89, 89, 85, 89, 89, 85, 85, 85, 89, 89
990 DATA 85, 89, 89, 101, 86, 89, 101, 101, 101, 89, 86, 85, 85, 85, 106, 106
1000 DATA 85, 106, 106, 85, 101, 89, 86, 86, 86, 89, 101, 85, 89, 102, 86, 89
1010 DATA 89, 85, 89, 85, 85, 85, 85, 170, 170, 85, 85, 85, 89, 89, 106, 106
1020 DATA 106, 89, 106, 85, 89, 89, 89, 89, 89, 89, 89, 89, 85, 85, 85, 170
1030 DATA 170, 85, 85, 85, 85, 85, 170, 170, 85, 85, 85, 85, 85, 170, 170, 85
1040 DATA 85, 85, 85, 85, 85, 85, 85, 85, 170, 170, 85, 85, 101, 101, 101, 101
1050 DATA 101, 101, 101, 101, 89, 89, 89, 89, 89, 89, 89, 89, 85, 85, 85, 165
1060 DATA 169, 105, 89, 89, 101, 101, 105, 106, 90, 85, 85, 85, 89, 89, 105
1070 DATA 105, 165, 165, 85, 85, 149, 149, 149, 149, 149, 149, 170, 170, 149
1080 DATA 149, 165, 101, 105, 89, 90, 86, 86, 86, 90, 89, 105, 101, 165, 149
1090 DATA 170, 170, 149, 149, 149, 149, 149, 149, 170, 170, 86, 86, 86, 86, 86
1100 DATA 86, 85, 105, 170, 170, 170, 170, 105, 85, 85, 85, 85, 85, 85, 170
1110 DATA 170, 85, 102, 106, 106, 106, 106, 89, 89, 85, 101, 101, 101, 101
1120 DATA 101, 101, 101, 101, 85, 85, 85, 90, 106, 105, 101, 101, 150, 150
1130 DATA 170, 105, 105, 170, 150, 150, 85, 105, 170, 150, 150, 170, 105, 85
1140 DATA 89, 89, 102, 102, 89, 89, 106, 85, 89, 89, 89, 89, 89, 89, 89, 89
1150 DATA 89, 89, 106, 106, 106, 89, 89, 85, 101, 101, 101, 170, 170, 101, 101
1160 DATA 101, 149, 149, 101, 101, 149, 149, 101, 101, 101, 101, 101, 101, 101
1170 DATA 101, 101, 101, 85, 85, 106, 106, 102, 102, 102, 85, 170, 170, 106
1180 DATA 106, 90, 90, 86, 86, 85, 85, 85, 85, 85, 85, 85, 85, 165, 165, 165
1190 DATA 165, 165, 165, 165, 165, 85, 85, 85, 85, 170, 170, 170, 170, 170, 85
1200 DATA 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 170, 149, 149
1210 DATA 149, 149, 149, 149, 149, 149, 153, 153, 102, 102, 153, 153, 102, 102
1220 DATA 86, 86, 86, 86, 86, 86, 86, 86, 85, 85, 85, 85, 153, 153, 102, 102
1230 DATA 170, 170, 169, 169, 165, 165, 149, 149, 86, 86, 86, 86, 86, 86, 86
1240 DATA 86, 101, 101, 101, 106, 106, 101, 101, 101, 85, 85, 85, 85, 90, 90
1250 DATA 90, 90, 101, 101, 101, 106, 106, 85, 85, 85, 85, 85, 85, 169, 169
1260 DATA 89, 89, 89, 85, 85, 85, 85, 85, 85, 170, 170, 85, 85, 85, 106, 106
1270 DATA 101, 101, 101, 101, 101, 101, 170, 170, 85, 85, 85, 85, 85, 85, 170
1280 DATA 170, 101, 101, 101, 101, 101, 101, 165, 165, 101, 101, 101, 149, 149
1290 DATA 149, 149, 149, 149, 149, 149, 149, 149, 149, 149, 149, 149, 149, 149
1300 DATA 86, 86, 86, 86, 86, 86, 86, 86, 170, 170, 85, 85, 85, 85, 85, 85
1310 DATA 170, 170, 170, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 170, 170, 170
1320 DATA 86, 86, 86, 86, 86, 86, 170, 170, 85, 85, 85, 85, 165, 165, 165, 165
1330 DATA 90, 90, 90, 90, 85, 85, 85, 85, 101, 101, 101, 165, 165, 85, 85, 85
1340 DATA 165, 165, 165, 165, 85, 85, 85, 85, 165, 165, 165, 165, 90, 90, 90
1350 DATA 90,-999
1360 DATA 120, 162, 25, 142, 20, 3, 162, 192, 142, 21, 3, 88, 96, 162, 49, 142
1370 DATA 20, 3, 162, 234, 142, 21, 3, 88, 96, 206, 56, 3, 240, 3, 76, 49, 234
1380 DATA 174, 55, 3, 142, 56, 3, 173, 54, 3, 106, 144, 12, 174, 52, 3, 172
1390 DATA 34, 208, 142, 34, 208, 140, 52, 3, 106, 144, 12, 172, 53, 3, 174, 35
1400 DATA 208, 140, 35, 208, 142, 53, 3, 76, 49, 234, 255,999
1405 READA:X=X+A:IFA=999THEN1420
1410 GOTO1400
1420 RESTORE:L=2048:IFX<>174658THENPRINT"ERROR IN DATA":STOP
1430 READA:IFA=-999THEN1450
1440 POKEL,A:L=L+1:GOTO1430
1450 POKE56333,127:POKE1,51:FORI=0TO2047:POKE4096+I,PEEK(53248+I)
1460 POKE1,55:POKE56333,129
1470 L=49152
1480 READA:IFA=999THEN1500
1490 POKEL,A:L=L+1:GOTO1480
1500 POKE53272,19:POKE53270,216
4000 FORI=0TO2048:READA:IFPEEK(2048+I)<>ATHENPRINTI:STOP
4010 NEXT
READY.
```



WILLIAM FONG

Rocky IV

Can you handle the keyboard with boxing gloves?



Everyone has heard of boxing superstar Rocky, as played so well by Sylvester Stallone in those great films. Now here's your chance to play the role from the comfort of your computing chair!

Your opponent in this prize fight is Apollo, the greatest fighter of all time. He's been trained by a computer and has never been beaten before. Your aim is to beat him and take his title.

Your controls in the ring are: @ to move forward; (punch with your left arm;) punch with your right arm;? step back. You have 15 rounds of 20 seconds each to knock your opponent to the ground.

When you RUN the program the screen will go black and then green as the title music appears. Now begin! The bell will ring to signify the start of round one and there is a display of time, rounds and points on screen.

Keep an eye on the amount of energy you use as this is important. If you use more than 500 points worth then you'll be so weak that you could be knocked out. The game continues until someone wins on points or a knockout. Seconds out. . !

VARIABLES USED

HI	*high value of note*
LO	*low value of note*
V	*beginning of VIC-chip*
VL	*volume address*
H,L	*high and low addresses for voice 1*
W	*waveform address for voice 1*
AT	*attack/delay address for voice 1*
TI$	*time string*
A	*value of depressed key*
C	*sprite collision*
R	*score for Rocky*
P	*score for Apollo*
K	*whether Apollo punches or moves*
T1	*energy for Rocky*
T2	*energy for Apollo*
X,Y	*positions of Rocky*
X1,Y1	*positions of Apollo*
U	*knock down routine*
Q	*number of round*

HOW IT RUNS

1	*screen black and message*
2-11	*dimension arrays and sprite DATA*
12	*sets values for tune*
13-14	*set up variables*
15-17	*graphic routines*
18-21	*clear screen, print title and display sprites*
23-24	*set up values and volume*
25-27	*print information for scores, etc*
28	*sets sprite pointer to normal*
29	*punching sound*
36-41	*move and punch*
42	*time check*
43-44	*check on knockout*
45-46	*make Rocky and Apollo jump*
47	*repeats main routine*
51-55	*end of round routine*
56-60	*start of round routine*
61	*check for hit on Rocky*
62-63	*count for Rocky*
67-68	*energy ckeck*
69-76	*knockout routine*
77-122	*sprite and tune DATA with memory start at 832 to 12798*
124	*position sprites*
125-127	*play tune*
128-130	*start the game*
133-138	*winning message*

```
0 REM *** ROCKY IV : WILLIAM FONG    **
1 POKE53281,0:POKE53280,0:PRINT"]■"TAB(14)"PLEASE WAIT"
2 DIMHI(100),LO(100):FORN=0TO62:READA:POKE12288+N,A:NEXT
3 REM **** POKES DATA FOR SPRITES ***
4 FORN=0TO62:READA:POKE12352+N,A:NEXT
5 FORN=0TO62:READA:POKE12416+N,A:NEXT
6 FORN=0TO62:READA:POKE12480+N,A:NEXT
7 FORN=0TO62:READA:POKE12544+N,A:NEXT
```

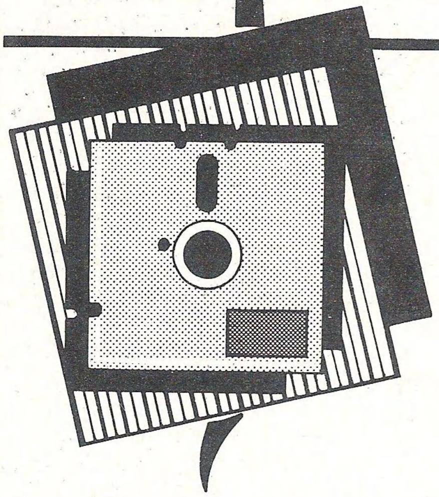
```
8 FORN=0TO62:READA:POKE12608+N,A:NEXT
9 FORN=0TO62:READA:POKE12672+N,A:NEXT
10 FORN=0TO62:READA:POKE12736+N,A:NEXT
11 FORN=0TO62:READA:POKE832+N,A:NEXT
12 FORN=0TO98:READHI(N),LO(N):NEXT
13 PRINT"▒":VL=54296:H=54273:L=54272:W=54276:AT=54277:Q=0:T1=10:T2=10
14 V=53248:X=170:Y=160:X1=120:TI$="000000"
15 POKEV+21,7:POKEV+23,7:POKEV+29,7:POKEV+32,13:POKEV+33,13
16 POKE2040,192:POKEV+27,7:POKE2041,195:POKEV+40,9
17 POKEV+39,8:POKEV+28,7:POKEV+37,1:POKEV+38,0
18 PRINT"▒■▒         ROCKY IV  : BY WILLIAM FONG          ■▒▒▒▒▒▒▒▒▒▒▒▒▒"
19 FORN=1TO4:PRINT"■         ▒▒──────────────────────────────■▒":NEXT
20 FORN=1TO3:PRINT"▒         ▒                                  ■":NEXT
21 POKEV+0,X:POKEV+1,Y:POKEV+2,X1:POKEV+3,Y:GOTO124
22 REM ****** MAIN ROUTINE **********
23 A=PEEK(197):C=PEEK(V+30):POKEVL,15
24 U=INT(RND(1)*5):K=INT(RND(1)*5)
25 PRINT"▒▒▒■▒ TIME  :"RIGHT$(TI$,3):PRINT"▒▒▒▒■▒ ROCKY   :"R:PRINT"■▒ APOLLO  :"IN
T(P)
26 PRINT"▒▒▒▒"TAB(20)"■▒ ENERGY USED  :"T1:T1=T1+2:T2=T2+2
27 PRINT"▒▒▒▒▒"TAB(20)"■▒ ENERGY USED  :"INT(T2):T2=T2-.2
28 POKEV+0,X:POKEV+1,Y:POKE2040,192:POKEV+2,X1:POKEV+3,Y:POKE2041,195:P=P+.3
29 IFC=3ANDA<>64THENPOKEW,0:POKEW,33:POKEH,1:POKEL,233:POKEAT,3
30 IFA=45THENPOKE2040,193
31 IFA=50THENPOKE2040,194
32 IFA=46ANDX>100THENX=X-5:IFX<X1+30THENX1=X1-5
33 IFA=55ANDX<250THENX=X+5
34 IFK=1THENPOKE2041,196
35 IFK=2THENPOKE2041,197
36 IFK=3ANDX1<230THENX1=X1+5:IFX1>X-30THENX=X+5
37 IFK=4ANDX1>65THENX1=X1-5
38 IFC=3ANDA=46ORA=55THENT1=T1+1
39 IFA=45ORA=50ORA=55ORA=46THENR=R+1:T1=T1+1
40 IFK=1ORK=2ORK=3ORK=4THENT2=T2+1
41 IFC=3ANDK=1ORK=2THENP=P+1:T2=T2+1
42 IFTI$>="000021"THEN49
43 IFU>2ANDC=3THEN62
44 IFU=1ANDC=3THEN70
45 IFT=0THENY=Y-1:Y1=Y1-2:T=1:GOTO47
46 IFT=1THENY=Y+1:Y1=Y1+2:T=0:GOTO47
47 GOTO23
48 REM **** END OF ROUND ********
49 POKEVL,15:FORN=1TO4:POKEW,0:POKEW,17:POKEH,96:POKEL,254:POKEAT,19
50 FORM=1TO200:NEXTM,N
51 IFX1>65THENX1=X1-3
52 IFX<250THENX=X+3
53 IFX>249ANDX1<66THEN56
54 POKEV+0,X:POKEV+2,X1
55 GOTO51
56 Q=Q+1:PRINT"▒▒■▒ ROUND  :"Q
57 IFQ>15THEN134
58 FORN=1TO300:NEXT:FORN=1TO35:X=X-2:X1=X1+2:POKEV+0,X:POKEV+1,Y
59 POKEV+2,X1:POKEV+3,Y:NEXT
60 T1=T1-10:T2=T2-10:TI$="000000":GOTO23
61 REM **** ROCKY KNOCKED OUT *******
62 IFK<>2ORT1<500THEN23
63 POKE2040,198:N=INT(T1/100):X=X+5:POKEV+0,X
64 FORM=1TON:FORL=0TO300:NEXTL:PRINT"▒■▒▒▒▒▒▒▒ ROCKY COUNT   :"M
65 POKEW,0:POKEW,33:POKEH,17:POKEL,37:POKEAT,56
66 NEXT:IFQ>=15THEN134
67 IFN>=10THENR=0:GOTO134
68 POKE2040,195:T1=T1+N:P=P+30:GOTO23
69 REM **** APOLLO KNOCKED OUT *****
70 IFT2<500ORA<>50THEN23
71 POKE2041,199:N=INT(T2/100):X1=X1-5:POKEV+2,X1
72 FORM=1TON:FORL=0TO300:NEXTL:PRINT"▒■▒▒▒▒▒▒▒ APOLLO COUNT  :"M
```

```
73 POKEW,0:POKEW,33:POKEH,17:POKEL,37:POKEAT,56
74 NEXT:IFQ>=15THEN134
75 IFN>=10THENP=0:GOTO134
76 POKE2041,196:T2=T2+N:R=R+30:GOTO23
77 REM ****** ROCKY NORMAL **********
78 DATA0,60,0,0,236,0,3,232,0,3,136,0,2,168,0,2,168,0,0,40,0,0,40,0
79 DATA0,40,0,0,84,0,0,84,0,0,136,0,2,136,0,2,8,0,2,10,0,2,2,128
80 DATA2,0,176,3,0,48,3,0,48,15,0,0,0,0,0
81 REM ***** ROCKY LEFT PUNCH ******
82 DATA0,60,0,0,44,0,0,232,0,0,200,0,234,168,0,202,168,0,0,40,0,0,40,0
83 DATA0,40,0,0,40,0,0,84,0,0,84,0,2,8,0,2,8,0,10,8,0,8,10,0
84 DATA8,10,192,8,0,192,12,0,192,12,0,0,60,0,0
85 REM ***** ROCKY RIGHT PUNCH ******
86 DATA0,60,0,0,44,0,0,40,0,234,136,0,202,170,0,0,42,0,0,42,0,0,58,0
87 DATA0,58,0,0,40,0,0,20,0,0,84,0,0,136,0,0,136,0,0,136,0,0,138,0
88 DATA0,130,128,0,128,176,0,192,48,0,192,48,3,192,0
89 REM ***** APOLLO NORMAL ********
90 DATA0,63,0,0,59,0,0,58,0,0,34,192,0,42,192,0,42,0,0,40,0,0,40,0
91 DATA0,40,0,0,40,0,0,20,0,0,21,0,0,34,0,0,34,0,0,34,0,0,34,0
92 DATA0,162,0,3,131,0,3,3,0,3,3,192,0,0,0
93 REM ***** APOLLO RIGHT PUNCH ****
94 DATA0,60,0,0,56,0,0,56,0,0,32,192,0,42,171,0,42,163,0,40,0,0,40,0,0,40,0
95 DATA0,40,0,0,40,0,0,20,0,0,21,0,0,34,0,0,34,128,0,32,128,0,32,32
96 DATA3,128,32,3,128,48,3,0,48,0,0,60
97 REM **** APOLLO LEFT PUNCH ******
98 DATA0,60,0,0,56,11,0,56,35,0,32,160,0,170,128,0,170,0,0,172,0,0,172,0
99 DATA0,160,0,0,160,0,0,80,0,1,80,0,2,32,0,2,32,0,2,32,0,2,32,0
100 DATA10,32,0,56,48,0,48,48,0,48,60,0,0,0,0
101 REM **** ROCKY OUT ***************
102 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
103 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,10,128,192,26,139,202,158,171,250
104 DATA158,175,0,0,0
105 REM **** APOLLO OUT ************
106 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
107 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,2,168,0,226,169,3,250
108 DATA173,163,250,173,175,0,0,0
109 REM **** WINNER ******************
110 DATA3,60,192,3,40,192,2,40,128,2,0,128,2,170,128,2,170,128,0,170,0,0,40,0
111 DATA0,40,0,0,40,0,0,40,0,0,20,0,0,22,0,0,34,0,0,34,0,0,34,0,0,34,0
112 DATA0,34,0,0,34,0,0,51,0,0,51,0
113 REM ***** TUNE ********
114 DATA21,154,21,154,25,177,21,154,17,37,19,63,17,37,21,154,19,63,0,0,19,63,19
115 DATA63,19,63,19,63,17,37,21,154,0,0,21,154,25,177,28,214,0,0,17,37,19,63,17
116 DATA37,21,154,19,63,0,0,19,63,19,63,19,63,19,63,17,37,21,154,0,0,19,63,0,0
117 DATA21,154,0,0,22,227,22,227,22,227,22,227,21,154,19,63,17,37,17,37,19,63
118 DATA21,154,19,63,0,0,19,63,0,0,21,154,22,227,22,227,22,227,22,227,21,154
119 DATA19,63,17,37,21,154,19,63,0,0,19,63,0,0,21,154,22,227,22,227,22,227,22
120 DATA227,21,154,19,63,17,37,17,37,19,63,21,154,19,63,0,0,19,63,0,0,21,154
121 DATA0,0,22,227,0,0,21,154,0,0,22,227,0,0,25,177,0,0,22,227,0,0,28,214,28
122 DATA214,0,0,19,63,17,37,19,63,17,37
123 REM*PLAYS TUNE "EYE OF THE TIGER"*
124 POKEV+0,X:POKEV+2,X1:POKEVL,2
125 FORN=0TO98:POKEW,0:POKEW,33:POKEH,HI(N):POKEL,LO(N):POKEAT,63
126 A=PEEK(197):IFA<>64THEN131
127 FORM=1TO110:NEXTM,N
128 PRINT"♥■◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙               ◪ PRESS ANY KEY TO START "
129 A=PEEK(197):IFA<>64THEN131
130 GOTO125
131 PRINT"♥■◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙                                      "
132 GOTO49
133 REM **** SORTS OUT WHO WON *******
134 IFR>PTHENPOKE2040,13:PRINT"]■◪ ROCKY YOU DID IT! ■"
135 IFP>RTHENPOKE2041,13:PRINT"]■◪ ROCKY YOU LOST! ■"
136 RUN2
137 A=PEEK(197):IFA=64THEN137
138 PRINTA:GOTO137
```

IAIN MURRAY

Tape Filer

Tape file on disc? Confused? You won't be... read on



Do you have lots of music tapes? Do you have trouble remembering which tune is where? If so, then this program is for you.

The program is in three sections. One allows you to create a disc file for five cassettes with tape length, title, date and all names and artists catalogued. The next allows you to correct or modify the files you've created. The last allows you to search quickly through all your files for a particular tune or artist and print up where this tune is to be found.

Although designed for disc users, the data could also be stored on cassette tape but searching would be very slow. But there's no need to restrict the use of the utility to music storage — try using the program as an index to your records or (most obviously) your tape and disc software!

Tune and artist are filed as a single string, separated by a 'spade' graphic — this you must insert yourself in the editor routine. Total line length should be kept below 78 characters.

The numbers in the DATA files are important to the running of the program and should not be removed or changed. Enter as 'UN' if you don't know the artist.

When searching with the reader section you can look for up to five names at once. A whole tape can also be viewed and the program will automatically search through all your DATA files, print out any findings and the location of that tune in your collection. It will stop if it finds anything but if you press SHIFT LOCK then it will search continuously until it reaches the last file, with the time taken printed at the end.

VARIABLES USED

AR$(10)	*artists' names*
TU$(10)	*tune names*
K2(10)	*number of references found*
AL(10)	*length of artist's name*
TL(10)	*length of tune name*
D$(1000)	*DATA lines*
M	*number of highest tape on file*
FM$	*current filename*
D	*tape number*
DA$	*input DATA line*
N3	*number of files to be searched for*
QT,F	*tape to be searched*
TT	*time*
TU	*tune number*
SI	*side number*
T$	*tune name or title*
R$	*artist's name*
L	*tape length*
D	*DATA counter*
S	*side counter*
C1,C2	*DATA counters*
K	*item to be edited*

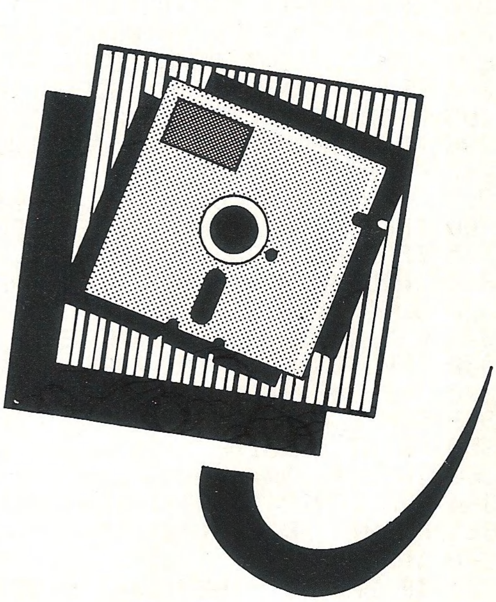
HOW IT RUNS

40-70	*set up and title*
80-90	*ask for highest tape on file*
100-135	*main menu selection*
150-220	*menu and choice selection*
230-280	*list complete tape*
294-395	*read in DATA, checking for interrupts*
400-480	*separate out and print tune and artist*
520-522	*input number of names to be searched for*
530-575	*input required names*
580-597	*input start tape number and make up filename*
599-670	*input DATA and take appropriate action*
680-690	*check for artist*
700-710	*check for tune*
720-726	*check for artist and tune*
730-777	*print artist, tune and location*
780-830	*go on to next tape if required*
840-850	*print number of references found and search time*
1000-1020	*input start tape number and make up filename*
1030-1070	*input tape length for five tapes*
1080-1130	*input side title and date for two sides*
1140-1220	*input tunes and artists*
1230-1240	*put tune and artists into single string*
1250-1360	*save DATA to disc*
2000-2015	*input start tape number and make up filename*
2020-2100	*load DATA*
2110-2210	*list DATA, checking for interrupts*
2220-2320	*menu and choice selection*
2330-2400	*resaving DATA file*
2900-2960	*trap disc errors and continue*
3000	*exit program*

```
10 REM *** TAPE FILER ***
20 REM *** BY IAIN MURRAY (C) 1984 ***
40 POKE 53280,6:POKE 53281,6
50 DIM AR$(10),TU$(10),K2(10),AL(10),TL(10),D$(1000)
59 REM [CRSR DOWN] [PINK] [WHITE] [CRSR DOWN]
60 SK=0:N$="▒▒THIS IS NOT ACCEPTABLE !▒▒"
```

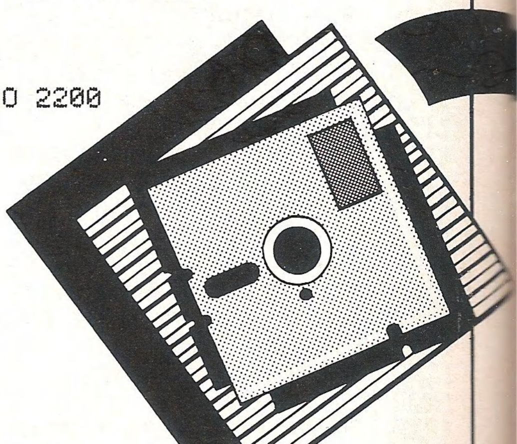
```
69 REM [CLR] [WHITE] [4 CRSR DOWN] [13 CRSR RIGHT] [RVS ON] [RVS OFF]
70 PRINT "♥EQQQQ]]]]]]]]]]]]]R MUSIC FILER ■"
79 REM [3 CRSR DOWN]
80 PRINT "QQQGIVE NUMBER OF HIGHEST TAPE ON":INPUT "FILE : ";M
90 IF M<0 OR M<>INT(M) OR M/5<>INT(M/5) THEN 80:REM ** CHECK M DIVISIBLE BY 5
98 REM *** MAIN MENU ***
99·REM [CLR] [3 CRSR DOWN]
100 PRINT "♥QQQYOU CAN :"
104 REM * EACH LINE STARTS WITH A [CRSR DOWN]
105 PRINT "Q1) READ A FILE"
110 PRINT "Q2) CREATE A FILE"
115 PRINT "Q3) EDIT A FILE"
120 PRINT "Q4) EXIT PROGRAM"
124 REM [2 CRSR DOWN]
125 PRINT "QQWHICH ONE DO YOU WANT (1-4) ?"
130 GET A$:A=VAL(A$):IF A<1 OR A>4 THEN 130
135 ON A GOTO 150,1000,2000,3000
148 REM [CLR] [3 CRSR DOWN]
149 REM *** READER SECTION ***
150 PRINT "♥QQQYOU CAN :"
159 REM * EACH LINE STARTS WITH A [CRSR DOWN]
160 PRINT "Q1) SEE A COMPLETE TAPE"
170 PRINT "Q2) FIND ALL TUNES BY A PARTICULAR ARTIST";
180 PRINT "Q3) FIND A PARTICULAR TUNE BY ANY ARTIST"
190 PRINT "Q4) FIND A PARTICULAR TUNE BY A            PARTICULAR ARTIST"
195 PRINT "Q5) RETURN TO MAIN MENU"
199 REM [2 CRSR DOWN]
200 PRINT "QQWHICH ONE DO YOU WANT (1-5) ?"
210 GET C$:C=VAL(C$):IF C<1 OR C>5 THEN 210
220 ON C GOTO 230,520,520,520,100
229 REM *** COMPLETE TAPE ***
230 PRINT "♥":REM [CLR]
235 INPUT "QEWHICH TAPE ";N:REM [CRSR DOWN] [WHITE]
240 IF N<1 OR N>M OR N<>INT(N) THEN PRINT N$:GOTO 235
285 N2=N
290 IF ((N2-1)/5)<>INT((N2-1)/5) THEN N2=N2-1:GOTO 290
291 FM$="TAPEFILE"+STR$(N2)+STR$(N2+4)
293 REM [2 CRSR DOWN] [WHITE] * 295 START WITH [CRSR DOWN]
294 PRINT "QQESEARCHING FOR TAPE #";N
295 PRINT  "QPRESS "CHR$(34)"S"CHR$(34)" TO INTERRUPT LISTING, "CHR$(34)"C";
296 PRINTCHR$(34)" TO  CONTINUE LISTING, OR "CHR$(34)"A"CHR$(34)" TO ABANDON"
297 PRINT"LISTING"
299 OPEN 1,8,15:OPEN 3,8,3,FM$+",S,R"
300 INPUT#1,A:IF A<>0 THEN 2900
305 IF N=N2 THEN 330
310 INPUT#3,DA$:IF DA$="3" THEN N2=N2+1:GOTO 305
320 GOTO 310
329 REM [CRSR DOWN] [WHITE]
330 INPUT#3,L$:PRINT "QEC-";L$
340 INPUT#3,S$
349 REM [CRSR DOWN] [CYAN]
350 PRINT "Q◣SIDE ";S$
360 INPUT#3,T$
364 REM [CRSR DOWN] [PINK] [CRSR DOWN]
365 PRINT "Q▒";T$;"Q"
370 INPUT#3,DA$
375 IF DA$="2" THEN S$=DA$:GOTO 350
380 IF DA$="3" THEN 455
385 GET A$:IF A$="A" THEN CLOSE 3:PRINT"QQLISTING ABANDONNED":GOTO 455
387 IF A$<>"S" THEN 400
390 GET A$:IF A$="A" THEN CLOSE 3:PRINT"QQLISTING ABANDONNED":GOTO 455
395 IF A$<>"C" THEN 390
400 FOR M2=1 TO LEN(DA$)
409 REM [SPADE] IS SHIFTED "A"
410 IF MID$(DA$,M2,1)="♠" THEN 430
420 NEXT M2
430 TU$=LEFT$(DA$,(M2-1))
434 REM [WHITE] [LIGHT BLUE]
```

```
435 AR$=RIGHT$(DA$,(LEN(DA$)-M2))
440 IF LEN(TU$)<20 AND LEN(AR$)<20 THEN PRINT "▣"TU$TAB(20)"◘"AR$:GOTO 370
444 REM [WHITE] [LIGHT BLUE]
445 PRINT "▣"TU$
446 PRINT "◘"TAB(5)AR$
450 GOTO 370
455 CLOSE 3:CLOSE 1
459 REM [2 CRSR DOWN] [WHITE] [RVS ON] [RVS OFF]
460 PRINT "▩▩▣PRESS ▨ SPACE ■ TO CONTINUE :"
470 GET A$:IF A$=" " THEN FOR I=1 TO 10:K2(I)=0:NEXT I:GOTO 150
480 GOTO 470
518 REM *** PARTICULARS WANTED ***
519 REM [CRSR DOWN]
520 PRINT "▩HOW MANY NAMES DO YOU WANT TO SEARCH    FOR (1-5) ?"
521 GET A$:IF A$<"1" OR A$>"5" THEN 521
522 N3=VAL(A$)
523 FOR N4=1 TO N3
527 IF C=3 THEN 560
529 REM [CRSR DOWN]
530 INPUT "▩ARTIST'S NAME ";AR$(N4):AL(N4)=LEN(AR$(N4))
540 IF AL(N4)<1 OR AL(N4)>39 THEN PRINT N$:GOTO 530
550 IF C=2 THEN 575
559 REM [CRSR DOWN]
560 INPUT "▩NAME OF TUNE ";TU$(N4):TL(N4)=LEN(TU$(N4))
570 IF TL(N4)<1 OR TL(N4)>39 THEN PRINT N$:GOTO 560
575 NEXT N4
579 REM [CRSR DOWN]
580 PRINT "▩GIVE NUMBER OF TAPE TO COMMENCE"
585 INPUT"SEARCH ";QT
590 IF QT<1 OR QT>M OR QT<>INT(QT) THEN PRINT N$:GOTO 580
592 IF ((QT-1)/5)<>INT((QT-1)/5) THEN QT=QT-1:GOTO 592
593 REM [CRSR DOWN] [RVS ON] [RVS OFF]
594 PRINT"▩USE ▨SHIFT LOCK■ FOR CONTINUOUS SEARCH":TT=TI
595 IF ((QT-1)/5)<>INT((QT-1)/5) THEN QT=QT-1:GOTO 595
596 IF QT>M THEN 840
597 FM$="TAPEFILE"+STR$(QT)+STR$(QT+4)
598 REM [2 CRSR DOWN]
599 PRINT "▩▩SEARCHING ";FM$
600 OPEN 1,8,15:OPEN 5,8,4,FM$+",S,R"
610 INPUT#1,A:IF A<>0 THEN 2900
620 T=QT:TU=0:SI=1
625 TU=TU+1
626 GOTO 630
627 INPUT#5,DA$
630 INPUT#5,DA$
635 IF DA$="1" THEN 627
640 IF DA$="2" THEN SI=2:TU=1:GOTO 627
650 IF DA$="3" THEN SI=1:T=T+1:TU=1
660 IF DA$="3" AND T=QT+5 THEN QT=QT+5:GOTO 820
665 IF DA$="3" THEN 627
670 ON C-1 GOTO 680,700,720
679 REM *** PARTICULAR ARTIST ***
680 FOR N5=1 TO N3
682 IF RIGHT$(DA$,AL(N5))=AR$(N5) THEN 730
685 NEXT N5
690 GOTO 625
699 REM *** PARTICULAR TUNE ***
700 FOR N5=1 TO N3
702 IF LEFT$(DA$,TL(N5))=TU$(N5) THEN 730
705 NEXT N5
710 GOTO 625
719 REM *** PARTICULAR TUNE AND ARTIST ***
720 FOR N5=1 TO N3
721 REM [SPADE] IS SHIFTED "A"
722 IF DA$=TU$(N5)+"♠"+AR$(N5) THEN 730
724 NEXT N5
726 GOTO 625
728 REM [SPADE] IS SHIFTED "A"
```

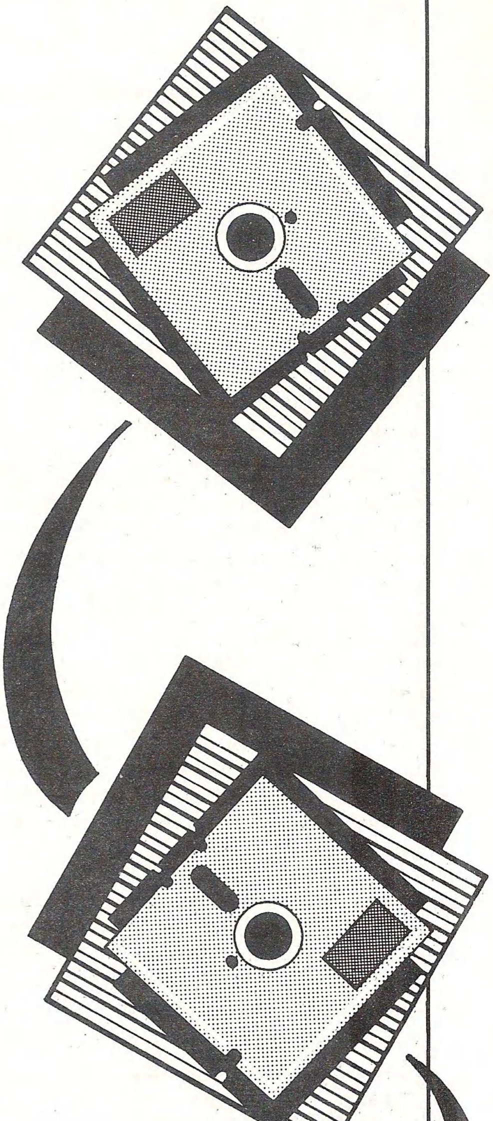


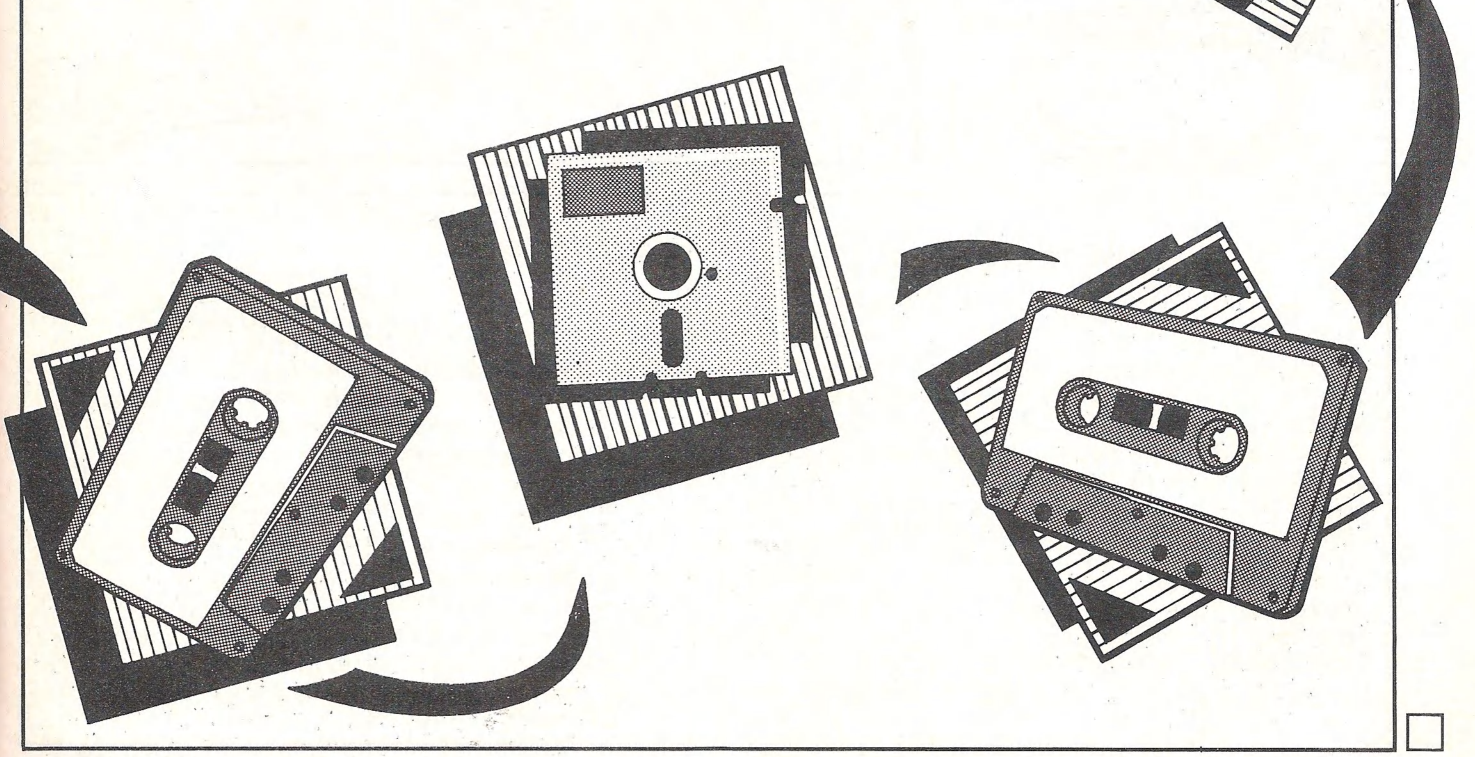
```
730 FOR I=1 TO LEN(DA$):IF MID$(DA$,I,1)="♠" THEN 750
740 NEXT I
750 T$=LEFT$(DA$,(I-1)):R$=RIGHT$(DA$,(LEN(DA$)-I)):K2(N5)=K2(N5)+1
755 IF R$<>"UN" THEN 760:REM [CRSR DOWN]
756 PRINT "■";T$;" BY AN UNKNOWN ARTIST"
757 GOTO 770
759 REM [CRSR DOWN]
760 PRINT "■";T$;" BY ";R$
770 PRINT "IS TUNE #";TU;"ON SIDE";SI;"OF TAPE #";T
775 IF PEEK(654)<>1 THEN 780:REM [CRSR DOWN]
776 PRINT "■SEARCHING: "
777 ON C-1 GOTO 685,705,724
779 REM [CRSR DOWN]
780 PRINT "■KEEP SEARCHING (Y/N) ?"
790 GET A$:IF A$="Y" THEN PRINT "■SEARCHING :":ON C-1 GOTO 685,705,724
800 IF A$<>"N" THEN 790
805 CLOSE 5:CLOSE 1
810 GOTO 150
820 REM *** GOING ON TO NEXT TAPE ***
825 CLOSE 5:CLOSE 1
830 GOTO 596
838 REM *** END OF FILES ***
839 REM [2 CRSR DOWN] [CRSR DOWN]
840 PRINT "■■LAST TAPE FILE SEARCHED■"
841 FOR N4=1 TO N3
842 ON C-1 GOTO 843,845,847:REM [CRSR DOWN] ON NEXT THREE LINES.
843 PRINT"■";K2(N4);"REFERENCES TO ";AR$(N4):PRINT "WERE FOUND":NEXT N4:GOTO 848
845 PRINT"■";K2(N4);"REFERENCES TO ";TU$(N4):PRINT "WERE FOUND":NEXT N4:GOTO 848
846 PRINT"■";K2(N4);"REFERENCES TO ";TU$(N4);" BY":PRINTAR$(N4);" WERE FOUND"
847 NEXT N4:REM [CRSR DOWN]
848 PRINT "■RUN TIME -";(INT((TI-TT)/60)/10)*10;" SECONDS"
849 IF PEEK(654)=1 THEN PRINT "■■REMOVE ■SHIFT■ NOW":REM[DOWN][PINK][RVSON,OFF]
850 GOTO 460
 998 REM *** CREATOR SECTION ***
 999 REM [CLR] [3 CRSR DOWN]
 1000 PRINT "■■■■GIVE NUMBER OF FIRST TAPE IN NEW FILE :":INPUT N
 1010 IF N<M OR N<>INT(N) OR ((N-1)/5)<>INT((N-1)/5) THEN 1000
 1020 W1$=STR$(N):W2$=STR$(N+4):FM$="TAPEFILE"+W1$+W2$:D=1
 1030 FOR I=N TO N+4:REM ** LOOP FOR 5 TAPES **
 1039 REM [WHITE] [4 CRSR DOWN]
 1040 PRINT "■■■■■TAPE NUMBER";I
 1049 REM [2 CRSR DOWN]
 1050 INPUT "■■GIVE TAPE LENGTH (IN MINUTES) ";L
 1060 IF L<0 OR L>120 OR L<>INT(L) THEN PRINT N$:GOTO 1050
 1070 D$(D)=STR$(L):D=D+1
 1080 FOR S=1 TO 2:REM ** LOOP FOR 2 SIDES **
 1090 D$(D)=STR$(S):D=D+1
 1099 REM [CRSR DOWN] [WHITE] [CRSR LEFT]
 1100 PRINT "■■TAPE";I;"(C -";L;"■) SIDE";S
 1109 REM [CRSR DOWN]
 1110 PRINT "■GIVE TITLE OF SIDE, AND DATE :"
 1120 INPUT T$:IF LEN(T$)<2 OR LEN(T$)>39 THEN PRINT N$:GOTO 1110
 1130 D$(D)=T$:D=D+1
 1139 REM [CRSR DOWN] [WHITE] [LIGHT BLUE] [WHITE] [LIGHT BLUE]
 1140 PRINT "■■GIVE NAME OF ■TUNE■, OR "CHR$(34)"X"CHR$(34)" TO END SIDE :■"
 1150 INPUT TU$
 1160 IF TU$="X" THEN 1240
 1170 IF LEN(TU$)>39 THEN PRINT N$:GOTO 1150:REM ** NAME TOO LONG **
 1179 REM [CRSR DOWN] [WHITE] [BLACK] [WHITE]
 1180 PRINT "■■GIVE NAME OF ■ARTIST■, OR "CHR$(34)"S"CHR$(34)" FOR PREVIOUS ";
 1189 REM [BLACK]
 1190 PRINT "ARTIST :■"
 1200 INPUT AS$
 1210 IF AS$="S" THEN AS$=AR$
 1220 IF LEN(AS$)>39 THEN PRINT N$:GOTO 1200:REM ** NAME TOO LONG **
 1229 REM [SPADE] IS SHIFTED "A"
 1230 D$(D)=TU$+"♠"+AS$:D=D+1:AR$=AS$:GOTO 1140
 1240 NEXT S:D$(D)="3":D=D+1:NEXT I:D=D-1
```

```
1248 REM *** SAVING ROUTINE ***
1249 REM [CLR] [WHITE] [3 CRSR DOWN]
1250 PRINT"■■■■■DATA INPUT COMPLETE"
1259 REM [CRSR DOWN] [RVS ON] [RVS OFF]
1260 PRINT "■INSERT DISK AND PRESS ■ SPACE ■ WHEN READYTO SAVE :"
1270 GET A$:IF A$<>" " THEN 1270
1279 REM [2 CRSR DOWN]
1280 PRINT "■■SAVING :"
1290 OPEN 1,8,15:OPEN 2,8,3,FM$+",S,W"
1300 INPUT#1,A:IF A<>0 THEN 2900
1310 FOR I=1 TO D:PRINT#2,D$(I):NEXT I:CLOSE 2:CLOSE 1
1319 REM [3 CRSR DOWN]
1320 PRINT "■■■SAVING COMPLETE"
1329 REM [CRSR DOWN] [RVS ON] [RVS OFF]
1330 PRINT "■PRESS ■ SPACE ■ TO CONTINUE :"
1340 GET A$:IF A$<>" " THEN 1340
1350 IF N+4>M THEN M=N+4
1360 GOTO 100
1998 REM *** EDIT SECTION ***
1999 REM [CLR]
2000 PRINT "■"
2004 REM [2 CRSR DOWN]
2005 PRINT "■■GIVE NUMBER OF FIRST TAPE IN FILE :":INPUT F
2010 IF F<0 OR F>M OR F<>INT(F)OR((F-1)/5)<>INT((F-1)/5) THEN PRINTN$:GOTO 2005
2015 F$="TAPEFILE"+STR$(F)+STR$(F+4)
2020 C=1:C2=0:OPEN 1,8,15:OPEN 2,8,3,F$+",S,R"
2030 INPUT#1,A:IF A<>0 THEN 2900
2050 INPUT#2,D$(C)
2059 REM [2 CRSR DOWN]
2060 IF C=1 THEN PRINT "■■LOADING ";F$
2070 IF D$(C)="3" THEN C2=C2+1:IF C2=5 THEN 2090
2080 C=C+1:GOTO 2050
2090 CLOSE 2:CLOSE 1
2099 REM [2 CRSR DOWN]
2100 PRINT "■■LOADING COMPLETE"
2109 REM [CRSR DOWN]
2110 PRINT "■PRESS "CHR$(34)"S"CHR$(34)" TO INTERRUPT LISTING, "CHR$(34)"C";
2112 PRINTCHR$(34)" TO  CONTINUE LISTING, OR "CHR$(34)"A"CHR$(34)" TO ABANDON"
2114 PRINT"LISTING."
2115 REM [CRSR DOWN]
2116 PRINT "■(NOTE THAT THE FILE CONTAINS NUMBERS AS WELL AS YOUR DATA.)"
2117 REM [CRSR DOWN] [RVS ON] [RVS OFF] [2 CRSR DOWN]
2118 PRINT "■PRESS ■ SPACE ■ WHEN READY :■■"
2120 GET A$:IF A$<>" " THEN 2120
2129 REM [RVS ON] [RVS OFF]
2130 FOR I=1 TO C:PRINT I,"■";D$(I);"■"
2139 REM [CRSR DOWN] [PINK] [WHITE]
2140 GET A$:IF A$="A" THEN PRINT "■■LISTING ABANDONNED■":GOTO 2200
2150 IF A$<>"S" THEN 2190
2160 GET A$:IF A$="C" THEN 2190
2169 REM [CRSR DOWN] [PINK] [WHITE]
2170 IF A$="A" THEN PRINT "■■LISTING ABANDONNED■":GOTO 2200
2180 GOTO 2160
2190 NEXT I
2199 REM [CRSR DOWN] [RVS ON] [RVS OFF]
2200 PRINT "■PRESS ■ SPACE ■ TO CONTINUE :"
2210 GET A$:IF A$<>" " THEN 2210
2219 REM [CLR] [3 CRSR DOWN]
2220 PRINT "■■■■YOU CAN :"
2229 REM [2 CRSR DOWN]
2230 PRINT "■■1) VIEW DATA"
2239 REM * EACH LINE STARTS WITH [CRSR DOWN]
2240 PRINT "■2) DELETE DATA ITEM"
2250 PRINT "■3) INSERT DATA ITEM"
2260 PRINT "■4) SWAP DATA ITEM"
2270 PRINT "■5) SAVE NEW DATA FILE"
2280 PRINT "■6) RETURN TO MAIN MENU"
2289 REM [2 CRSR DOWN]
```



```
2290 PRINT "▒▒WHICH DO YOU WANT (1-6) ?"
2300 GET A$:A=VAL(A$):IF A<1 OR A>6 THEN 2300
2320 ON A GOTO 2110,2330,2330,2330,2410,100
2329 REM [2 CRSR DOWN] [CRSR LEFT]
2330 PRINT "▒▒WHICH ITEM (1 -";C;"▌) : ";:INPUT K
2340 IF K<1 OR K>C THEN PRINT N$:GOTO 340
2350 ON A-1 GOTO 2360,2370,2380
2359 REM *** DELETE ITEM ***
2360 FOR I=K TO C:D$(I)=D$(I+1):NEXT I:C=C-1:GOTO 2220
2369 REM *** INSERT ITEM ***
2370 FOR I=C TO K STEP -1:D$(I+1)=D$(I):NEXT I:C=C+1
2379 REM [CRSR DOWN]
2380 INPUT "▒NEW ITEM : ";D$(K)
2390 IF LEN(D$(K))>78 THEN PRINT N$:GOTO 2380
2400 GOTO 2220
2408 REM *** RESAVING ROUTINE ***
2409 REM [2 CRSR DOWN]
2410 PRINT "▒▒RESAVING ";F$
2420 OPEN 1,8,15,"S:"+F$:CLOSE 1
2430 OPEN 1,8,3,F$+",S,W"
2440 FOR I=1 TO C:PRINT#1,D$(I):NEXT I:CLOSE 1
2449 REM [2 CRSR DOWN]
2450 PRINT "▒▒RESAVING COMPLETE"
2459 REM [CRSR DOWN] [RVS ON] [RVS OFF]
2460 PRINT "▒PRESS ◩ SPACE ■ TO CONTINUE :"
2470 GET A$:IF A$<>" " THEN 2470
2480 GOTO 100
2899 REM *** DISK ERROR HANDLING ***
2900 CLOSE 1:CLOSE 2:CLOSE 3:CLOSE 4:CLOSE 5
2910 OPEN 1,8,15,"I0":CLOSE 1
2929 REM [4 CRSR DOWN] [PINK]
2930 PRINT "▒▒▒▒▓REQUIRED FILE NOT FOUND ON DISK !"
2939 REM [WHITE] [2 CRSR DOWN] [RVS ON] [RVS OFF]
2940 PRINT "≡▒▒PRESS ◩ SPACE ■ TO CONTINUE :"
2950 GET A$:IF A$<>" " THEN 2950
2960 GOTO 100
2998 REM *** EXIT PROGRAM ***
2999 REM [CLR] [24 CRSR DOWN]
3000 PRINT "♥▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":END
```





JAMIE CLYDE

Music Maker

Learn how easy making music can become.

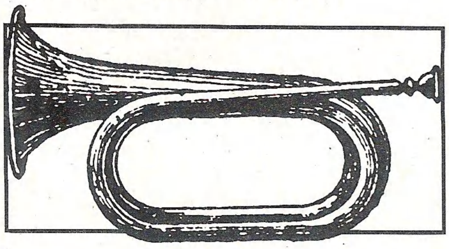
One of the strengths of the Commodore 64 is the ability to produce sound. However, beginners often feel that getting music out of the machine is next to impossible but it is really quite simple.

In order to produce a single note, you must turn on the speaker and set the waveform and pitch. A small amount of programming is necessary.

To turn on the speaker POKE 54296,15. Setting the wavelength is necessary then to tell the computer how to play the note and a choice of four is available: Triangle for a smooth, flowing note (POKE 54276,17); Sawtooth for a rough, broken note (POKE 54276,33); Pulse (POKE 54276,65); Sound/noise (POKE 54276,129).

To set the pitch of the note, the high and low frequency registers must be POKEd with the values given in the User Manual. For example, to play middle C, enter POKE 54273;34:POKE 54272,75.

The program demonstrates the above remarks. The object is to put a tune into the computer and then hear it played back. There is a choice of instruments and, if desired, backing music also.



HOW IT RUNS

100-130	*set up frequencies*
210-270	*main play routine*
500-550	*end of routine*
2000-2050	*play note*
3000-3450	*play backing music*
5000-5990	*set up instrument and other choices*
6000-6080	*input sound frequencies*
7000-7090	*set up sound settings*
7100-7170	*end of tune routine*
7500-7520	*play back tune*
8000-8140	*instructions*
10000-10070	*note DATA*

VARIABLES USED

NF	*note frequencies for tune*
SF	*set values for notes*
O1	*octave value to be added*
OC	*octave number*
I	*flag — solo or not*
K$	*keyboard string (valid keys)*
A1,A2	*backing music note values*
D	*tempo*
AD	*attack/delay*
SR	*sustain/release*
RT	*backing waveform*
WA	*tune waveform*
PH	*high pulse frequency*
PL	*low pulse frequency*
L1,L2	*frequency for backing music*
AT$	*cursor down string*
Q	*number of notes in tune*
R$	*get $*

```
0 REM *****************
1 REM *  MUSIC MAKER  *
2 REM *               *
3 REM *FOR COMMODORE64*
4 REM *               *
5 REM *      B Y      *
6 REM * JAMIE   CLYDE *
7 REM *[C] J.C APR.'83*
8 REM *****************
90 PRINT"█ HANG ON PLEASE!
100 DIMNF(500,2),SF(96,2),A1(2,2),A2(6,2),K$(21)
110 FORT=0TO94:READSF(T,1),SF(T,2):NEXT
120 K$="AWSEDFTGYHUJKOLP:;*=↑"
130 FORT=1TO21:K$(T)=MID$(K$,T,1):NEXT
200 GOSUB5000:O1=13*5
210 FORN=1TO500
220 GETR$:IFR$=""THENFORI=1TO100:NEXT:GOTO500
230 FORG=1TO21:IFK$(G)=R$THENGOSUB2000:GOTO530
240 NEXTG:IFR$="█"THENQ=N:GOTO7100
250 R=ASC(R$):IFR<133ORR>139THENR$="":GOTO500
260 OC=R-133:PRINT"█";LEFT$(AT$,2);TAB(15)"▓OCTAVE:█";OC
270 O1=13*OC
499 REM * FINNISH NOTE & RECORD IT *
500 IFR$=""THENNF(N,1)=0:NF(N,2)=0:POKE54283,0
530 IFA=1THENGOSUB3000
550 NEXTN
998 Q=N-1:GOTO7100
1999 REM * PLAY NOTE *
2000 IFO1+G>94THENG=G-21
2002 FORP=1TO2:NF(N,P)=SF(G+O1,P):NEXT
2010 POKE54283,WA
2020 FORKL=1TO2:POKE54281-KL,NF(N,KL):NEXT
2050 RETURN
2999 REM * VOICE NO.2 *
3000 ON F GOSUB3020,3200,3100
3010 GOTO3400
3020 L1=(NF(N,1)-13):L2=(NF(N,2)-13)
3025 IFL1<0THENL1=0
3027 IFL2<0THENL2=0
3030 RETURN
3100 AF=AF+1:IFAF=6THENAF=1
3110 L1=A2(AF,1):L2=A2(AF,2)
3120 RETURN
3200 AF=AF+1:IFAF=2THENAF=1
3210 L1=A1(AF,1):L2=A1(AF,2)
3250 RETURN
3400 POKE54272,L2:POKE54273,L1
3450 RETURN
4999 REM * SET UP PROGRAM *
5000 PRINT"█":POKE53280,3:POKE53281,6
5010 AT$="████████████████████"
5020 PRINTLEFT$(AT$,5);TAB(15)"█ MUSIC MAKE █
5030 PRINTTAB(12)"██(C)BY J. CLYDE '83"
5040 INPUT"████████DO YOU WANT INSTRUCTIONS [Y/N]";I$
5050 IFI$="Y"THENGOSUB8000
5060 PRINT"███ INPUT:
5070 PRINT"█████████1] PLAY INSTRUMENT
5080 PRINT"█████████2] DEFINE OWN SOUND
5090 PRINT"ENTER 1/2"
5100 INPUTI:IFI<>1ANDI<>2THEN5100
5110 ON I GOSUB5800,6000
5120 PRINT"███ INPUT: █
```

```
5130 PRINT"████████1] SOLO
5140 PRINT"████████2] ACCOMPANIED
5150 PRINT"ENTER 1/2
5160 INPUTI:IFI<>1ANDI<>2THEN5160
5170 IFI=2THENGOSUB5500
5180 PRINT"█":POKE53280,8:POKE53281,0
5190 GOSUB7000
5200 PRINT"█";TAB(13)"█ MUSIC MAKER █
5210 PRINT"███KEYS 'F1-F7'TO CHANGE OCTAVE
5220 PRINT"███CLR/HOME TO ESCAPE/PLAYBACK██
5230 PRINT"███W ██E ███T ██Y ██U ███O ██P ███* ██↑ █
5235 PRINT"█ A  S   DF   G   H   JK   L   :;   =   █
5240 PRINT"█";LEFT$(AT$,2);TAB(15)"█OCTAVE:5
5245 PRINT"█";LEFT$(AT$,18);TAB(10)"█ SOUND INFORMATION █
5250 PRINT"██WAVEFORM :";WA
5260 PRINT"█ATTACK/DECAY :";AD
5270 PRINT"█SUSTAIN/RELEASE :";SR
5280 PRINT"█PULSE H :";PH
5290 PRINT"█PULSE L :";PL
5490 RETURN
5499 REM * SET NON SOLO MODE *
5500 PRINT"██ INPUT: █
5510 PRINT"███████1] MELODIC  ACCOM.
5520 PRINT"███████2] CONSTANT ACCOM.
5530 PRINT"███████3] JAZZ     ACCOM.
5540 PRINT"INPUT 1/2/3
5550 INPUTI:IFI<>1ANDI<>2ANDI<>3THEN5550
5560 ON I GOSUB5600,5650,5700
5590 A=1:RETURN
5600 F=I:RT=17
5620 RETURN
5650 F=I:FORT=1TO2:A1(1,T)=SF(20,T):NEXT
5670 FORT=1TO2:A1(2,T)=SF(0,T):NEXT
5690 RT=33:RETURN
5700 F=I:F$="030834753685"
5710 FORTK=1TO3:A2(TK,1)=VAL(MID$(F$,TK*4,2))
5720 A2(TK,2)=VAL(MID$(F$,TK*4+2,2))
5730 NEXT:RT=129
5750 RETURN
5799 REM * CHOOSE AN INSTRUMENT *
5800 PRINT"██INPUT:█
5810 PRINT"██████1] PIANO
5820 PRINT"██████2] FLUTE
5830 PRINT"██████3] HARPSICORD
5840 PRINT"██████4] XYLOPHONE
5850 PRINT"██████5] ORGAN
5860 PRINT"██████6] COLLIAPE
5880 PRINT"██████7] ACCORDIAN
5890 PRINT"██████8] TRUMPET
5900 PRINT"ENTER [1-8]"
5910 INPUTI:IFI<1ANDI>8THEN5910
5915 INPUT"██TEMPO (1TO50)";D
5920 D=20:IFI=1THENWA=65:AD=9:SR=0:PL=255:PH=0:RETURN
5930 IFI=2THENWA=17:AD=96:SR=0:RETURN
5940 IFI=3THENWA=33:AD=9:SR=0:RETURN
5950 IFI=4THENWA=17:AD=9:SR=0:RETURN
5955 IFI=5ORI=6THENWA=17:AD=0:SR=240:RETURN
5960 IFI=7THENWA=17:AD=102:SR=0:RETURN
5970 WA=33:AD=96:SR=0
5990 RETURN
5999 REM * DEFINE SOUND *
6000 INPUT"WAVEFORM [P/S/N/T]";W$
```

```
6010 INPUT"ATTACK/DECAY [0-255]";AD
6015 IFAD<0ORAD>255THEN6010
6020 INPUT"SUSTAIN/RELEASE [0-255]";SR
6025 IFSR<0ORSR>255THEN6020
6030 INPUT"TEMPO [1-50]";D
6040 IFW$="P"THENINPUT"PULSE(H [0-15],L [0-255])";PH,PL:WA=65
6050 IFW$="S"THENWA=33
6060 IFW$="T"THENWA=17
6070 IFW$="N"THENWA=129
6080 RETURN
6999 REM * SET SOUND SETTINGS *
7000 POKE54272,0:POKE54273,0
7010 POKE54279,0:POKE54280,0
7015 POKE54296,15:POKE650,0
7020 POKE54276,RT:POKE54283,WA
7030 POKE54277,64:POKE54284,AD
7040 POKE54278,00:POKE54285,SR
7050 IFWA=65THENPOKE54281,PL:POKE54282,PH
7090 RETURN
7099 REM * END OF PROGRAM ROUTINE *
7100 PRINT"▓▓▓ PRESS 'P' TO PLAY BACK
7110 PRINT"▓      'T' TO PLAY AGAIN
7120 PRINT"▓      'S' TO STOP
7130 INPUTR$
7140 IFR$="P"THENGOSUB7500:GOTO7100
7150 IFR$="S"THENPRINT"▓":POKE54296,0:END
7160 IFR$="T"THENPOKE54296,0:RUN100
7170 GOTO7130
7499 REM * PLAY BACK *
7500 GOSUB7000:FORN=1TOQ
7510 IFA=1THENGOSUB3000
7515 GOSUB2010
7520 FORT=1TOD:NEXTT,N:RETURN
7999 REM * INSTRUCTIONS *
8000 PRINT"▓";CHR$(8)CHR$(14)
8010 PRINTTAB(10)"▓▓▓ ╲ ╭ ♥ ╮ ─ ╲ ♠ ╯ ▔ ▁ ▓
8020 PRINTTAB(13)"▓ BY ╰AMIE ┤LYDE
8030 PRINT"▓▓▓▓▓│HE  OBJECT  OF  THIS PROGRAM  IS TO  ENABLE  YOU TO PLAY A";
8040 PRINT" TUNE AND HEAR IT  PLAYED  BACK TO YOU .  ▌DU  ARE ABLE TO  PLAY A ";
8050 PRINT" PREDEFINED  INSTRUMENT OR  YOU  CAN MAKE YOUR OWN SOUNDS YOURSELF.
8060 PRINT"▓▓▓▓▓╮F  YOU  CHOOSE  TO DESIGN  YOUR OWN  SOUND  THEN  YOU ";
8070 PRINT" WILL  BE  ASKED WHAT  WAVEFORM,ATTAGK &  DECAY AND SUSTAIN";
8080 PRINT" &  RELEASE.¬NTER THE WAVEFORM AS A LETTER  [IE. 'P'FOR PULSE]."
8090 PRINT"▓▓▓▓▓ ▌DU   CAN  BE   ACCOMPANIED  BY  THE  COMPUTER  IN  THREE  DIF";
8100 PRINT"FERENT  WAYS .  ╲ELODIC  MEANS  ";
8105 PRINT"THE  COMPUTER PLAYS AN  OCTAVE BELOW YOU . ";
8110 PRINT"┤ONSTANT  IS A ECHO  AND ╰AZZ IS A JAZZY BEAT!.
8120 PRINT"▓▓▓  ┐└▔♥♥  │┌  ♥┐♠─▔┤│♠▁  │┌  ┬╱│╮╱╭▔   ▓
8130 GETR$:IFR$<>" "THEN8130
8140 PRINTCHR$(142):RETURN
10000 DATA1,18,1,35,1,52,1,70,1,90,1,110,1,132,1,155,1,179,1,205,1,233,2,6
10010 DATA2,37,2,69,2,104,2,140,2,179,2,220,3,8,3,54,3,103,3,155,3,210
10020 DATA4,12,4,73,4,139,4,208,5,25,5,103,5,185,6,16,6,108,6,206,7,53,7,163,8,2
3
10030 DATA8,147,9,21,9,159,10,60,10,205,11,114,12,32,12,216,13,156,14,107,15,70
10040 DATA16,47,17,37,18,42,19,63,20,100,21,154,22,227,24,63,25,177,27,56
10050 DATA28,214,30,141,32,94,34,75,36,85,38,126,40,200,43,52,45,198,48,127
10060 DATA51,97,54,111,57,172,61,126,64,188,68,149,72,169,76,252,81,161,86,105
10065 DATA91,140,96,254,102,194,108,223,115,88,122,52,129,120,137,43,145,83
10070 DATA153,247,163,31,172,210,183,25,193,252,205,133,217,189,230,176,244,103

READY.
```

☐

IAIN MURRAY

Sprite Dumper

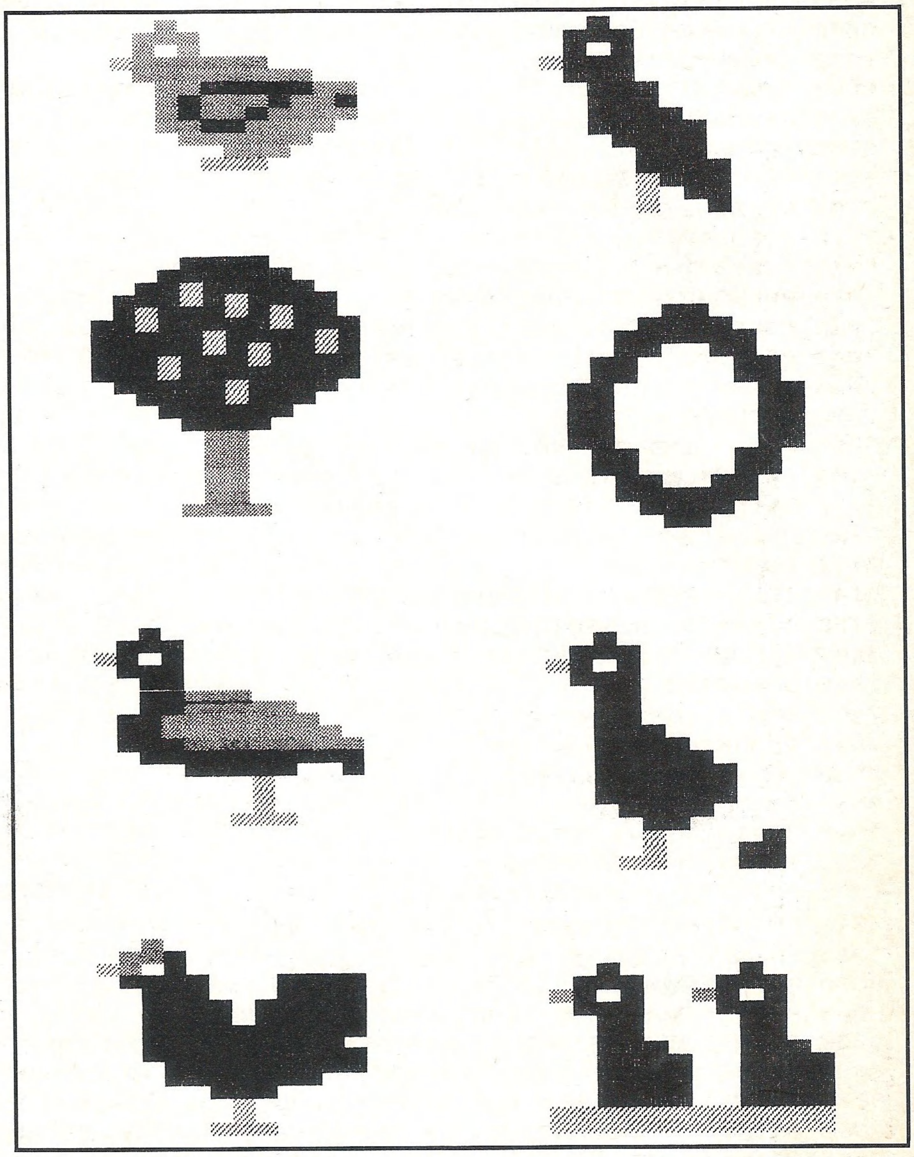
Print out your spritely creations with this routine.

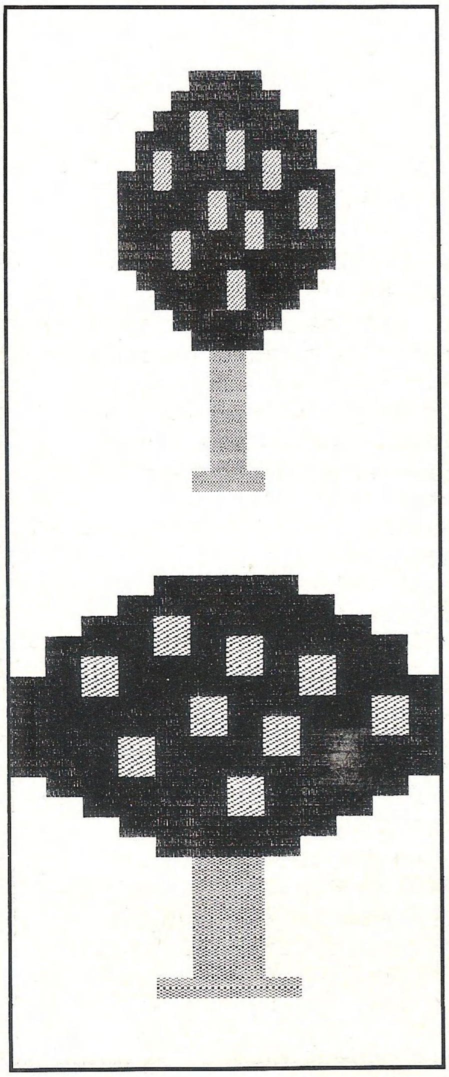
Programs to dump the contents of the 64's screen to a printer are fairly common, but sprites need a special routine because of the way they are stored in memory.

This is such a routine and will work directly on a 1525 graphic printer. It should also work with some modification on a 1515 or 1520, but see your printer manual for the relevant details.

The sprite data is read directly from its place in memory. It should be in the first block of memory (address 0-16384) but, if you are using sprites from another block, you merely need to alter the memory address (S) at line 94. The program then breaks the sprite down into its component parts (or bit pairs for multi-coloured sprites) and prints the corresponding shading pattern.

You also have the option of expanding the sprite facilities along either or both the X,Y directions.



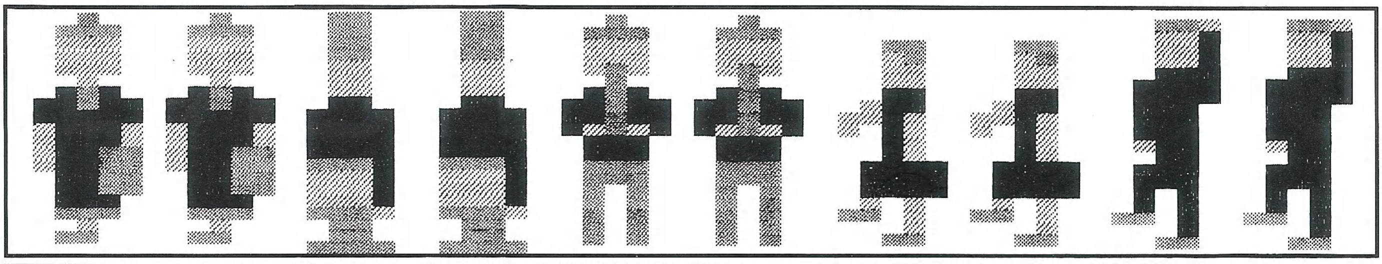


HOW IT RUNS

30-40	titles and open file to printer
50-105	input printing details
110-165	single-colour sprites
112-113	print blank line if needed
115-130	convert sprite data to bits
140-165	print shading as needed
170-245	as above, for multi-coloured sprites

VARIABLES USED

T$,D$,H$	prompt answers
S1	sprite block number (0-255)
S	sprite start address
D	double width flag
H	double height flag
I,J	sprite data row, column
K,C1,C2	bit types



```
10 REM *** SRITE PRINTER ***
20 REM *** BY IAIN MURRAY 1984 ***
28 REM *** PRINT TITLES ***
29 REM [CLR] [2 CRSR DWN] [10 SPACES] [WHITE] [REV ON] [REV OFF]
30 POKE 53280,6:POKE 53281,6:PRINT "▒▒▒          ▒▒ SPRITE PRINTER ■"
39 REM *** OPEN FILE TO PRINTER ***
40 OPEN 1,4:PRINT#1:G$=CHR$(8):N$=CHR$(15)
49 REM [2 CRSR DWN]
50 PRINT "▒▒NORMAL OR MULTI-COLOURED SPRITE, OR QUITPROGRAM (N/M/Q) ? ";
60 GET T$:IF T$<>"N" AND T$<>"M" AND T$<>"Q" THEN 60
69 REM [7 CRSR DWN]
70 PRINT T$:IF T$="Q" THEN PRINT "▒▒▒▒▒▒▒":PRINT#1,N$:CLOSE 1:END
79 REM *** GIVE MEMORY BLOCK NUMBER, NOT ADDRESS (BLOCK NUMBER=ADDRESS/64) ***
80 INPUT "▒GIVE BLOCK NUMBER OF SPRITE : ";S1
90 IF S1<0 OR S1>255 OR S1<>INT(S1) THEN 80
94 S=S1*64:REM ** THIS IS FOR 1ST 16K BLOCK OF MEMORY
95 REM [CRSR DWN]
96 PRINT "▒DOUBLE WIDTH PRINTING (Y/N) ? ";
97 GET D$:IF D$<>"Y" AND D$<>"N" THEN 97
98 PRINT D$:IF D$="Y" THEN PRINT#1,CHR$(14);
99 REM *** SET W$ FOR DOUBLE WIDTH PRINTING IF SELECTED ***
100 W$="":IF D$="Y" THEN W$=CHR$(14)
101 REM [CRSR DWN]
102 PRINT "▒DOUBLE HEIGHT PRINTING (Y/N) ? ";
103 GET H$:IF H$<>"Y" AND H$<>"N" THEN 103
105 PRINT H$:IF T$="M" THEN 170
109 REM *** SINGLE-COLOUR SPRITES ***
110 FOR I=0 TO 20:D=1:IF H$="N" THEN D=0:REM *** 21 ROWS PER SPRITE ***
111 REM ** CHECK FOR BLANK LINE **
112 IF PEEK(S+(3*I))<>0 OR PEEK(S+(3*I)+1)<>0 OR PEEK(S+(3*I)+2)<>0 THEN 115
113 PRINT#1:GOTO 165:REM ** TO SAVE PAPER, REMOVE PRINT#1 FROM THIS LINE
115 PRINT#1,N$"         "G$;
120 FOR J=0 TO 2:P=PEEK(S+J+(3*I)):REM *** 3 COLUMNS PER ROW ***
130 FOR K=7 TO 0 STEP -1:C=(2↑K) AND P:IF C=0 THEN 150:REM ** 8 BITS PER COLUMN
138 REM *** PRINT BLOCK OR SPACE ***
139 REM [REV ON] [SPACE] [REV OFF]
140 PRINT#1,N$W$"▒ ■";G$;:GOTO 160
150 PRINT#1,N$W$" ";G$;
159 REM *** DO LINE AGAIN IF DOUBLE HEIGHT SELECTED ***
160 NEXT K,J:PRINT#1:IF D>0 THEN D=D-1:GOTO 115
164 REM *** PRINT BLANK LINES ***
165 NEXT I:PRINT#1:PRINT#1:PRINT#1:GOTO 50
169 REM *** MULTI-COLOUR SPRITES ***
170 FOR I=0 TO 20:D=1:IF H$="N" THEN D=0:REM *** 21 ROWS PER SPRITE ***
171 REM ** CHECK FOR BLANK LINE **
172 IF PEEK(S+(3*I))<>0 OR PEEK(S+(3*I)+1)<>0 OR PEEK(S+(3*I)+2)<>0 THEN 175
173 PRINT#1:GOTO 245:REM ** TO SAVE PAPER, REMOVE PRINT#1 FROM THIS LINE
175 PRINT#1,N$"         "G$;
180 FOR J=0 TO 2:P=PEEK(S+J+(3*I)):REM *** 3 COLUMNS PER ROW ***
190 FOR K=7 TO 1 STEP -2:C1=(2↑K) AND P:C2=(2↑(K-1)) AND P:REM * 8 BITS PER COL
199 REM *** PRINT REQUIRED PATTERN FOR MULTI-COLOUR SPRITES ***
200 IF C1=0 AND C2=0 THEN PRINT#1,N$W$"  "G$;:GOTO 240
210 IF C1=0 THEN PRINT#1,N$W$"▒▒"G$;:GOTO 240
219 REM [REV ON] [REV OFF]
220 IF C2=0 THEN PRINT#1,N$W$"▒  ■"G$;:GOTO 240
230 PRINT#1,N$W$CHR$(17)"▀▀"CHR$(145)G$;
240 NEXT K,J:PRINT#1:IF D>0 THEN D=D-1:GOTO 175
245 NEXT I:PRINT#1:PRINT#1:PRINT#1:PRINT#1:GOTO 50
```

☐

BLOCKBUSTERS

Blockbusters is the ratings-topping, button-popping, mind-stopping quiz game of the decade, and now it's been adapted for your micro computer!

Central TV's popular programme attracts around 12 million fans, and now it's all set to become number one in the personal computer charts as well! Blockbusters is 'edutainment' of the highest level for all the family, providing a perfect introduction to the home computer.

Two players pit their wits against each other and the clock in a game which tests mental agility and general knowledge – it's exciting, educational and intense with an appeal as wide as the micro itself.

If you've got a computer in the family you've got to get Blockbusters. And if you haven't, it's worth getting one just to play Blockbusters! Now available from High Street stockists or direct from Macsen Software – just fill in the coupon below and try to contain your excitement – we'll deliver within 14 days!

This is what the experts have to say . . . "A surprisingly addictive family game with considerable educational side-benefits" – CRASH MICRO MAGAZINE

Macsen Software, GBA Limited, 17 Nott
Square, Carmarthen, Dyfed, SA3 1PQ.
Please send me a Blockbusters programme for my
BBC-B ELECTRON ☐ **SPECTRUM 48K** ☐
COMMODORE 64 ☐ (Tick appropriate box)
Price **£7.95** + **60** pence p&p.
I enclose Cheque/PO number __________
for **£8.55** made out to Macsen Software or please debit
my ACCESS/BARCLAYCARD, number __________
Signature (Mr/Mrs/Miss) __________
Address __________
____
__________ Postcode __________

THE ULTIMATE CHALLENGE!

MACSEN SOFTWARE

MACSEN SOFTWARE